Ph.D. in Computer Science

UNIVERSITY OF ALMERÍA

Department of Informatics

Quantum Computing and HPC Techniques for Solving Microrheology and Dimensionality Reduction Problems

Almería, October 2021

Author: Francisco José Orts Gómez

Supervisors:
Gracia Ester Martín Garzón
Gloria Ortega López

Quantum Computing and HPC Techniques for Solving Microrheology and Dimensionality Reduction Problems

Computación Cuántica y Técnicas HPC para Resolver Problemas de Micro-reología y de Reducción de Dimensionalidad

Almería, October 2021

Almería, Octubre 2021

Author/ Autor: Francisco José Orts Gómez

Supervisors/ Directoras:
Gracia Ester Martín Garzón
Gloria Ortega López

A mi hijo, Francisco José.

Agradecimientos (Acknowledgments)

En primer lugar, quiero mostrar mi agradecimiento a mis directoras, Ester y Gloria. Incluiros a vosotras en los agradecimientos me resulta extraño puesto que sois tan responsables como yo de esta tesis y no quiero que nadie os vea como una ayuda extra que he tenido, sino que sois un pilar fundamental de la tesis. En cualquier caso, quiero aprovechar para daros las gracias por todo. Hemos alcanzado muchos logros, muchos más de los que yo habría pensando al comenzar la tesis, y este documento es el testimonio escrito de lo que hemos logrado juntos.

En segundo lugar, quiero darle las gracias a mi compañero y amigo Nicolás Calvo Cruz por todo el soporte, profesional y personal, que me ha dado durante todo el desarrollo de la tesis. Cualquier cosa que diga de ti aquí no será suficiente para expresarte lo agradecido que estoy por todo, así que simplemente te voy a recordar que eres el mejor informático que he tenido la suerte de conocer desde que empecé a estudiar informática en un ya lejano 2002. Muchas gracias.

Y cómo no, mi infinito agradecimiento a todos mis coautores, personas todas ellas expertas en sus respectivas materias, de las que he aprendido mucho y sigo aprendiendo, y que me han hecho el favor de permitirme trabajar a su lado. Al profesor Antonio Puertas, cuyo trabajo fue el que me animó a comenzar mi tesis. A la profesora Inmaculada García, que siempre sabe decirnos el mejor camino a seguir. A los profesores Ernestas Filatovas y Olga Kurosova, siempre tan abiertos a explorar nuevas líneas y que se atreven con todo. Y al profesor Elías F. Combarro, mi maestro y ejemplo a seguir en computación cuántica.

Finalmente, mi agradecimiento al Ministerio de Ciencia, Innovación y Universidades por el contrato predoctoral para mi formación de doctor. Sin esta ayuda, no habría podido dedicarme en cuerpo y alma a la tesis y no habría podido llegar tan lejos.

Prefacio

La evolución de la tecnología siempre ha tenido como objetivo permitir que la sociedad pueda resolver sus problemas y mejorar su calidad de vida. En partícular, la informática es probablemente la ciencia que más ha evolucionado a partir de la segunda mitad del siglo XX, permitiendo alcanzar soluciones a problemas cada más desafiantes y variados gracias a su naturaleza interdisciplinar.

Esta tesis proporciona soluciones en disciplinas tan diferentes como el campo de la Microrreología, la reducción de dimensionalidad, y el tratamiento digital de imágenes. Como su nombre indica, esta tesis aborda la resolución de dichos problemas mediante el uso de computación cuántica y de computación de alto rendimiento, HPC. El rápido progreso que la computación cuántica está teniendo en los últimos años permite que podamos ya utilizarla no solo a nivel teórico, sino también práctico para resolver problemas actuales.

El propósito de esta tesis es doble. Primero, y como objetivo principal, se presende resolver utilizando computación cuántica y/o HPC una serie de problemas definidos formalmente en los campos previamente mencionados. Como se verá en la tesis, algunas de las soluciones aportadas podrían extrapolarse a otros campos de aplicación. Sin embargo, en esta tesis los exponemos en el contexto en que nosotros los hemos encontrado y resuelto. El segundo propósito de la tesis es el de contribuir al desarrollo de la computación cuántica de forma práctica. La aplicabilidad de este paradigma de computación tiene un importante cuello de botella causado por la escasez de recursos de los computadores cuánticos actuales y de las altas exigencias computacionales de los simuladores cuánticos. En esta tesis se estudian con detalle las formas de reducir dicha problemática, y se proponen soluciones en forma de circuitos que permitirán extender el uso de la computación cuántica a problemas de mayor complejidad.

El documento está dividido en cuatro capítulos. Puesto que ambos paradigmas de computación (cuántico y clásico) son utilizados para el mismo fin de resolver ciertos problemas, la estructura del documento está enfocada en diferenciar esos problemas, de forma que todos los esfuerzos orientados a un mismo resultado se presenten de forma sinérgica, no como alternativas que compiten entre sí.

El primer capítulo presenta los objetivos que persigue la tesis y el contexto en el que se desarrolla. Se presentan con profundidad (pero sin caer en repeticiones del contenido presentado en los artículos científicos que componen la tesis) las tecnologías utilizadas a lo largo de la tesis, para que el lector pueda tener una perspectiva global de los distintos aspectos abordados. También se explican nuestras contribuciones en el diseño de circuitos cuánticos, que han permitido que hayamos podido aplicar con éxito este paradigma de computación en nuestros problemas. Y, finalmente, se presentan los problemas que ha abordado la tesis y en los que hemos aplicado todo el conocimiento anteriormente explicado.

El segundo capítulo expone los artículos que componen esta tesis. Son un total de 10 artículos, 9 de ellos ya publicados en revistas reconocidas de alto impacto, y uno de ellos actualmente en proceso de revisión. Para su mejor comprensión, el capítulo está dividido en secciones que abordan, cada una, uno de los problemas tratados en la tesis. De esta forma, en una misma sección se pueden encontrar todas las soluciones que hemos aplicado en un mismo campo. Soluciones que, como norma, son totalmente complementarias.

El tercer capítulo hace un listado extenso de todas las aportaciones científicas realizadas como parte de la tesis. En una tesis por compendio se exige un mínimmo de artículos publicados en revistas de alto índice científico. Estos méritos son los que presentamos en el Capítulo 2. Sin embargo, nuestras publicaciones no se limitan a estos artículos, sino que incluyen un elevado número de aportaciones a congresos, jornadas, simposios, revistas sin índice de impacto, revisiones para revistas, y trabajos docentes.

El último capítulo presenta las conclusiones que hemos obtenido del trabajo realizado en la tesis. Se analiza el desarrollo de los objetivos de este trabajo y las claves tanto a nivel computacional como desde el punto de vista de cada aplicación. Se destacan tambien algunas limitaciones de las soluciones aportadas y se definen los trabajos futuros que pretendemos abordar.

Terminamos este prefacio describiendo de forma resumida la estructura del presente documento:

- El primer capítulo resume el trabajo realizado en la tesis.
- En el segundo capítulo se muestran las publicaciones que componen la tesis.
- El tercer capítulo enumera las aportaciones ciéntificas no incluidas en el segundo capítulo.
- Finalmente, el cuarto capítulo presenta las conclusiones.

Preface

The evolution of technology has always been aimed at enabling society to solve its problems and to improve its quality of life. In particular, computer science is probably the science that has evolved the most since the second half of the twentieth century, allowing us to reach solutions to increasingly challenging and varied problems thanks to its interdisciplinary nature.

This thesis provides solutions in very different disciplines such as the field of Microrheology, dimensionality reduction, and digital image processing. As its name indicates, this thesis addresses the resolution of such problems through the use of quantum computation and high performance computing. The rapid progress that quantum computation has been making in the last years allows us to use it not only at a theoretical level but also practical to solve current problems.

The purpose of this thesis is twofold. First, and as a main objective, it is intended to solve a series of formally defined problems, previously mentioned. As will be seen in the thesis, some of these solutions may well be extrapolated to other fields of application. However, in this thesis we present them in the context in which we have encountered and solved them. The second purpose of this thesis is to contribute to the development of quantum computing in a practical way. The applicability of this computing paradigm has an important bottleneck caused by the scarcity of resources of current quantum computers and the high computational demands of quantum simulators. In this thesis we study in detail the ways to reduce this problem, and we propose solutions in the form of circuits that will extend the use of quantum computing to larger problems.

The document is divided into four chapters. Since both paradigms (quantum and classical computing) are used for the same purpose of solving certain problems, the structure is focused on differentiating those problems, so that all efforts towards the same result are presented synergistically, not as competing alternatives.

The first chapter presents the objectives of the thesis and the context in which it is developed. The technologies used throughout the thesis are presented in depth (but without repeating the content presented in the scientific articles that make up the thesis), so that the reader can have a global perspective of its contents. It also explains our contributions in the design of quantum circuits, which have made it possible for us to successfully apply this computing paradigm to our problems. And finally, the problems addressed by the thesis and in which we have applied all the knowledge previously explained are presented.

The second chapter presents the articles that make up this thesis. There are a total of 10 articles, 9 of them already published in recognized high impact journals, and one of them is currently under review. Since it would be chaotic to present the articles individually or in chronological order based on their date of publication, the chapter is divided into sections, each of which deals with one of the problems addressed in the thesis. In this way, all the solutions we have applied in the same field can be found in the same section. Solutions that, as a rule, are fully complementary.

The third chapter makes an extensive listing of all the scientific contributions made as part of the thesis. A thesis by compendium requires a minimum number of articles published in journals with a high scientific index. However, our publications are not limited to these articles, but include a large number of contributions to congresses, conferences, journals without impact index, reviews for journals, and teaching papers.

The last chapter presents the conclusions we have drawn from the work done in the thesis. The development of the objectives of this work and the key computational and application aspects are discussed. We also highlight some limitations of the solutions provided and define the future work we intend to undertake.

We conclude this preface by summarizing the structure of this document:

- The first chapter summarizes the work carried out in the thesis.
- The second chapter shows the publications that make up this thesis.
- The third chapter lists the scientific contributions not included in the second chapter.
- Finally, the fourth chapter presents the conclusions.

Resumen

Esta tesis por compendio recoge un conjunto de diez trabajos. Dichos trabajos son el fruto de numerosas colaboraciones con diferentes grupos nacionales e internacionales, y están enfocados en resolver una serie de problemas de áreas tan distintas como son la micro-reología, el escalado multidimensional, y el tratamiento digital de imágenes. El punto en común de estos trabajos es que todos ellos utilizan la computación de alto rendimiento (HPC) y la computación cuántica como herramientas para solucionar los mencionados problemas. Estos dos paradigmas de computación no se utilizan como dos alternativas excluyentes la una de la otra, sino como enfoques totalmente complementarios que permiten optimizar aún más las soluciones alcanzadas por cada uno de los paradigmas por separado. En el caso de la micro-reología, se consigue extender mediante HPC un modelo físico ya existente, así como acelerar aún más la parte más costosa del cálculo involucrado en la extensión de dicho modelo mediante computación cuántica. En el caso del escalado multidimensional, los resultados obtenidos están enfocados a permitir que una de las técnicas de reducción más costosa en términos computacionales sea viable en tiempo y consumo energético. Finalmente, en el campo del tratamiento digital de imágenes hemos aplicado con éxito la técnica anterior como parte de un algoritmo mayor, llamado ISOMAP, que está enfocado en la clasificación de cierto tipo de imágenes. A la par, hemos obtenido un primer prototipo de binarizador para computadores cuánticos que mejora a los existentes en la literatura, como primer paso en nuestra carrera por construir un circuito mayor que nos permita mejorar ciertas partes de ISOMAP utilizando el paradigma cuántico. Una aportación importante de estos trabajos viene de la mano de los numerosos diseños de circuitos cuánticos propuestos, que optimizan a los disponibles en el estado del arte. Además, como complemento de estos 10 trabajos principales, y dentro de las tres áreas citadas, también se expone la lista de las más de 30 aportaciones a congresos nacionales e internacionales que se han llevado a cabo como parte de la tesis.

Abstract

This thesis by compendium gathers a set of ten papers. These works are the result of our numerous collaborations with different national and international groups, and are focused on solving a series of problems in areas as different as microrheology, multidimensional scaling, and digital image processing. The common point of these works is that they all use high-performance computing (HPC) and quantum computing as tools to solve these problems. These two computing paradigms are not used as two mutually exclusive alternatives, but as fully complementary approaches that further optimize the solutions achieved by each of the paradigms separately. In the case of microrheology, HPC is used to extend an existing physical model, as well as to further accelerate the most expensive part of the computation involved in the extension of that model by using quantum computation. In the case of multidimensional scaling, the obtained results are focused on allowing one of the most computationally expensive reduction techniques to be viable in terms of time and energy consumption. Finally, in the field of digital image processing we have successfully applied the above technique as part of a larger algorithm called ISOMAP that is focused on the classification of certain types of images. At the same time we have obtained a first prototype of a binarizator for quantum computers that improves the existing ones in the literature as a first step in our race to build a larger circuit that allows us to improve certain parts of ISOMAP using the quantum paradigm. An important contribution of this work comes from the numerous quantum circuit designs proposed, which optimize those available in the state of the art. Also, as a complement to these 10 main papers and within the three areas mentioned above, the list of more than 30 contributions to national and international congresses is presented.

Contents

List of Figures

Abbreviations used in Chapter 1

CPU	Central Processing Unit(s)
CUDA	Compute Unified Device Architecture
EE	Energy Efficiency
GA	Genetic Algorithm(s)
GPU	Graphics Processing Unit(s)
HSI	HyperSpectral Image(s)
HPC	High Performance Computing
ISOMAP	ISOmetric MAPping
KNN	K-Nearest Neighbor
MDS	MultiDimensional Scaling
MIMD	Multiple Instruction, Multiple Data
MPI	Message Passing Interface
NEQR	Novel Enhanced Quantum Representation
NUMA	Non-Uniform Memory Access
OpenCL	Open Computing Language
OpenMP	Open Multi-Processing
PThreads	POSIX Threads
QC	Quantum Computing
SIMD	Single-Instruction, Multiple Data
SM	Streaming Multiprocessor(s)
SMACOF	MAjorizing a COmplicated Function
TBB	Threading Building Blocks
UMA	Uniform Memory Access

1. Introduction

This chapter gives the reader a complete overview of the work done. It begins in Section 1.1 by justifying the current interest in High Performance Computing (HPC) techniques and quantum computing to extend and to accelerate specific computationally demand problems. It also sets out the main objectives to be achieved. Secondly, Section 1.2 gives a brief description of the specific HPC and quantum technologies used in this thesis. Section 1.3 presents the advances achieved in the thesis in terms of quantum circuits. Finally, Section 1.4 describes the problems covered, which are related to the fields of Microrheology, MultiDimensional Scaling, and Image Processing.

1.1 Context, motivation and goals

Problems that involve the processing of a large amount of data, as well as complex data structures and operations, are good candidates for taking advantage of the HPC techniques. Such techniques are based on the optimal parallel execution on modern computers, according to the specific combination of problem/ HPC platform. This parallelization offers important benefits, such as the resolution of big problems in less time and the ability to solve larger problems [34]. In this way, HPC allows interested researchers to extend their models and accelerate their simulations by exploiting a wide variety of computing resources. Some applications that typically benefit from these advantages and which are addressed in this thesis include physics simulations [46], dimensionality reduction of large volumes of data [23], and image processing [21]. However, the selection of HPC architectures and programming interfaces for the models to be developed requires significant ingenuity and is far from trivial [30].

Parallelizing involves a thorough understanding of the problem, as well as the ability to break it down into smaller problems that are relatively independent [31]. In an ideal case, the independence between subproblems is total. However, there are usually dependencies and constraints that complicate parallelization. Of course, it is also needed to understand the available technologies for parallelization. It is worth mentioning that each technology has its benefits and drawbacks, which are enhanced or diminished depending on the problem to which they are applied and the machines available to solve it [44]. In certain programming paradigms, such as GPU programming, special hardware constraints must be taken into account, such as reduced shared memory, libraries that work in a substantially different way than those we would use for multicore parallelization, or inter-GPU communications that are not intuitive for an outsider [66].

On the other hand, parallelizing a problem is not the only way to optimize its resolution. Choosing the best machines for the computation, applying the most efficient methodology according to the available resources, and even modelling the problem in other terms can allow solving a problem in less time or with fewer resources. In a world composed of such heterogeneous architectures and so many different tools, the possibilities to solve a problem are enormous. But as it usually happens when you have many possibilities, not all of them are equally good and it is necessary to have a strategy to choose the optimal one in terms of performance or energy consumption.

HPC is commonly used to try to improve the performance of the resolution of a large number of problems. In this thesis, HPC techniques are used to solve certain specific problems in Microrheology, dimensionality reduction (using a specific technique called SMACOF), and digital image processing. Moreover, quantum computing is also addressed to try to solve these problems. This thesis is not a comparison between HPC and quantum computing, but both computing paradigms are used in a complementary way and with a common purpose. Quantum computing is currently experiencing strong growth thanks mainly to the emergence of simulators and real prototypes freely available to the general public. With the proven and promising advantages offered by this type of computing [18, 63], it has been a focus in our thesis.

To develop the scientific models addressed in this thesis, the collaboration with other researchers has been essential. This way, we have been involved in several interdisciplinary work-teams. We want to emphasize the high degree of collaboration we have had with other departments and universities. It is fair to point out all these researchers have been a relevant key to define and model most of the applications addressed in this thesis. Next, we highlight the main researchers related to this thesis and their lines and institutions.

At the University of Almería itself, Department of Applied Physics, Dr. Antonio Puertas is the main responsible for the microrheology model treated in this thesis. Professor Matthias Fuchs, from the Physics Department of the Universität Konstanz, has also been essential in the extension of the model. At a computational level, we have also worked with Dr. Inmaculada García of the Department of Computer Science at the University of Málaga on the parallelization of suchs models. It has been the union of the knowledge of all these people that has made it possible to achieve results in this area.

The collaboration with the Computer Science Department of the University of Vilnius has given us important results in terms of dimensionality reduction and image processing. In particular, Dr. Ernestas Filatovas and Dr. Olga Kurasova have worked with us in this line from the very beginning, contributing with their wide experience in this and other topics. Professor Ernestas is also working with us in our line on quantum computation, which I will just talk about next.

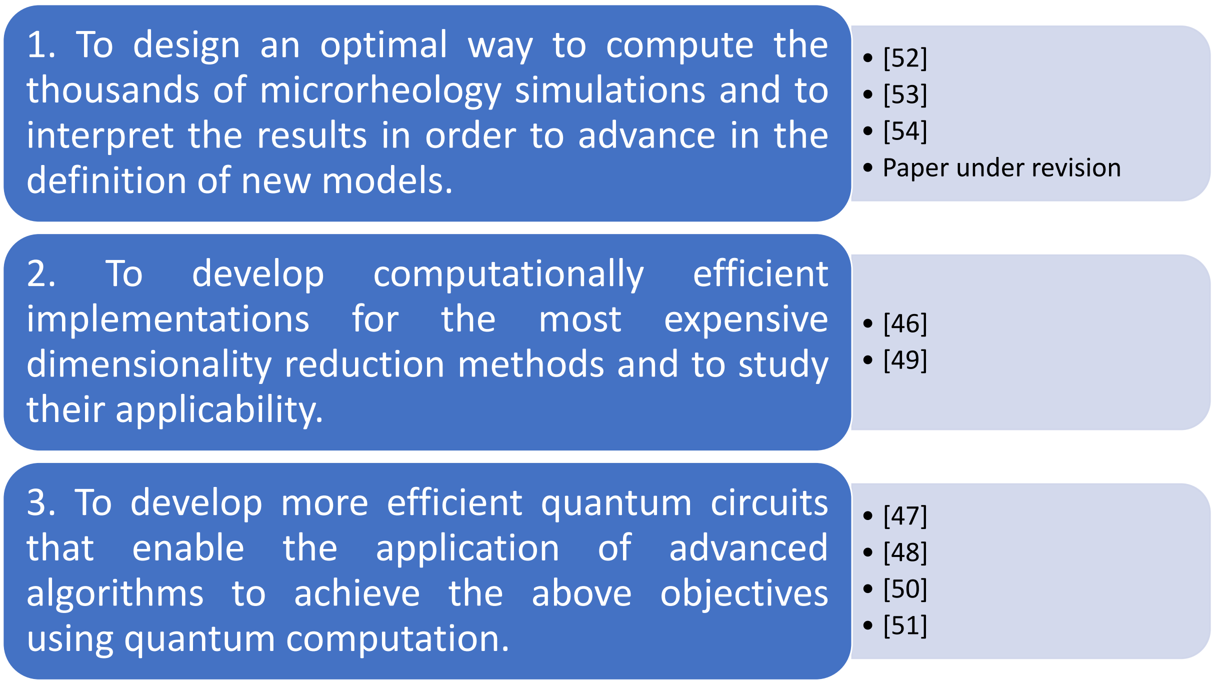
In quantum computing, Dr. Elías Combarro from the Computer Science Department of the University of Oviedo has been the one who has guided us the most and with whom we have achieved our best results. We have also worked in this line with Dr. Iñaki Fernández from the Mathematics Department of the University of Oviedo, who has provided the necessary statistics to advance in the definition of our problems in terms of quantum computation.

The problems that these people have raised have helped us to apply HPC and quantum computing in different contexts. They have also presented us with new computational challenges that have led to the extension of the corresponding models as well as to the development of some resources associated with those challenges.

As a consequence of the collaborations described above, the objectives addressed in the thesis were as follows:

- To design an optimal way to compute the thousands of microrheology simulations and to interpret the results in order to advance in the definition of new models. To extend the initial model, it is necessary to perform a large number of simulations, so it is necessary to study the possibilities -in terms of hardware, techniques, etc.- to perform this computation in as short a time as possible. Apart from the large number of simulations to be carried out, each one involves a heavy computational load. The approach should seek the best way to solve one simulation, and then focus on finding the best way to solve all of them.
- To develop computationally efficient implementations for the most expensive dimensionality reduction methods and to study their applicability. The most accurate dimensionality reduction methods are, unfortunately, the most expensive. Their applicability is very limited precisely because of the high computational costs involved. Their use should be extended to allow their application to real problems of high dimensions that can benefit from their high accuracy. As a complement, study possible applications to reduce costs and consumption of this type of methods. Moreover, we want to study the applicability of dimensionality reduction methods in hyperspectral image classification, as well as the general improvement of the accuracy of such classification.
- To analyse the applicability of quantum computation in the previous objectives, and to develop more efficient quantum circuits that enable the application of advanced algorithms to achieve the above objectives using quantum computation. Despite the great achievements made in quantum computing at the theoretical level, the truth is that at the practical level it is still in its infancy. The scarcity of resources of current quantum computers prevents the implementation of solutions for medium and large problems. We aim to study the applicability of quantum computing in the above points, so searching for the best circuits in the literature and even trying to design more efficient circuits also become objectives if we want to be able to implement the corresponding algorithms on today's small quantum computers.

To illustrate the different contributions of this thesis in several journals of international impact, Figure 1.1 summarizes the journal articles we have published as a result of addressing these objectives.



1. To design an optimal way to compute the thousands of microrheology simulations and to interpret the results in order to advance in the definition of new models.
- [52]
- [53]
- [54]
- Paper under revision

2. To develop computationally efficient implementations for the most expensive dimensionality reduction methods and to study their applicability.
- [46]
- [49]

3. To develop more efficient quantum circuits that enable the application of advanced algorithms to achieve the above objectives using quantum computation.
- [47]
- [48]
- [50]
- [51]

Figure 1.1: Thesis objectives and published papers as a result of addressing these objectives.

1.2 Technologies

This thesis has been carried out using classical and quantum computing technologies. The term "classical computing" arises to distinguish non-quantum computing from quantum computing. In particular, in this thesis classical computing involves HPC techniques related to the current supercomputers, i.e. distributed architecture of nodes of multi-core processors which are accelerated by Graphics Processing Units (GPUs).

1.2.1 Classical HPC technologies

The previous section discussed the usefulness of dividing a large problem into a set of smaller problems in order to work in parallel with them. Splitting a problem into other problems involves not only properly separating the instructions that make up the problem, but also separating the data. Moreover, the available hardware to solve a problem is a determining factor in choosing the appropriate HPC technique to solve it. However, although the hardware sets the limits of what can and cannot be done, it is essential to know this hardware in depth in order to make good use of the features it offers [9].

The most widespread classification of parallel computers is Flynn's taxonomy. This classification distinguishes four architectures according to the number of instructions and data streams. In particular, we are interested in two of these architectures: Single-Instruction, Multiple Data (SIMD), and Multiple Instruction, Multiple Data (MIMD).In the first type, the processors execute the same instruction but concurrently access different data, the data blocks being usually handled as vectors. In the second type, processors handle global parallelism of both instructions and data.

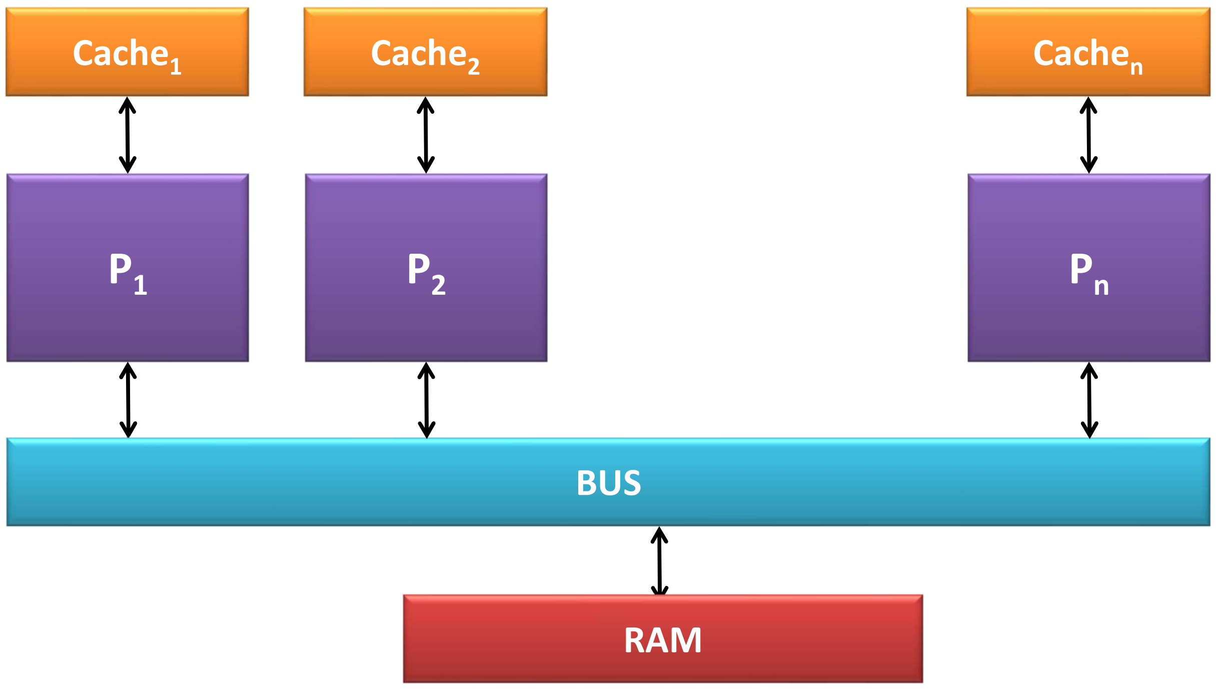


Figure 1.2: Example diagram of a multicore architecture.

In this, MIMD is the most widely used architecture. MIMD in turn can be divided into two groups according to its memory architecture. A first group would consist of those systems in which the processors access the same memory space. A second type would include systems whose processors have different memory modules assigned to them, so they must communicate via messages even though they are part of the same logical system. The first type, the so-called multiprocessors, are simpler to use because of their directly shared information. However, they also have drawbacks. Firstly, it is the programmer who is entirely responsible for synchronization. Secondly, there is a much more difficult problem to solve: the scalability between memory and processors is very limited.

Multicore

Instead of focusing on increasing clock frequency or processor speed, adding more processors (cores) to the same chip has become the architecture of choice in terms of improving overall system performance. Such cores are connected to the same bus, sharing the same memory device and other resources of the chip, as well as all external resources. This is shown in Figure 1.2. In terms of memory, current implementations are vendor-dependent, and include shared or independent cache modules, bus implementations and extra threading capabilities, etc.

Depending on its internal structure, multiprocessors can be divided into two other types: centralized and distributed shared memory. The former are called Uniform Memory Access (UMA) because the access time of all processors to all memory is precisely the same. The latter consists of individual nodes with their own resources but connected through an interconnection network (Figure 1.3). In this way, processors can access their local memory faster than any other. This is why they are called Non-Uniform Memory Access (NUMA).

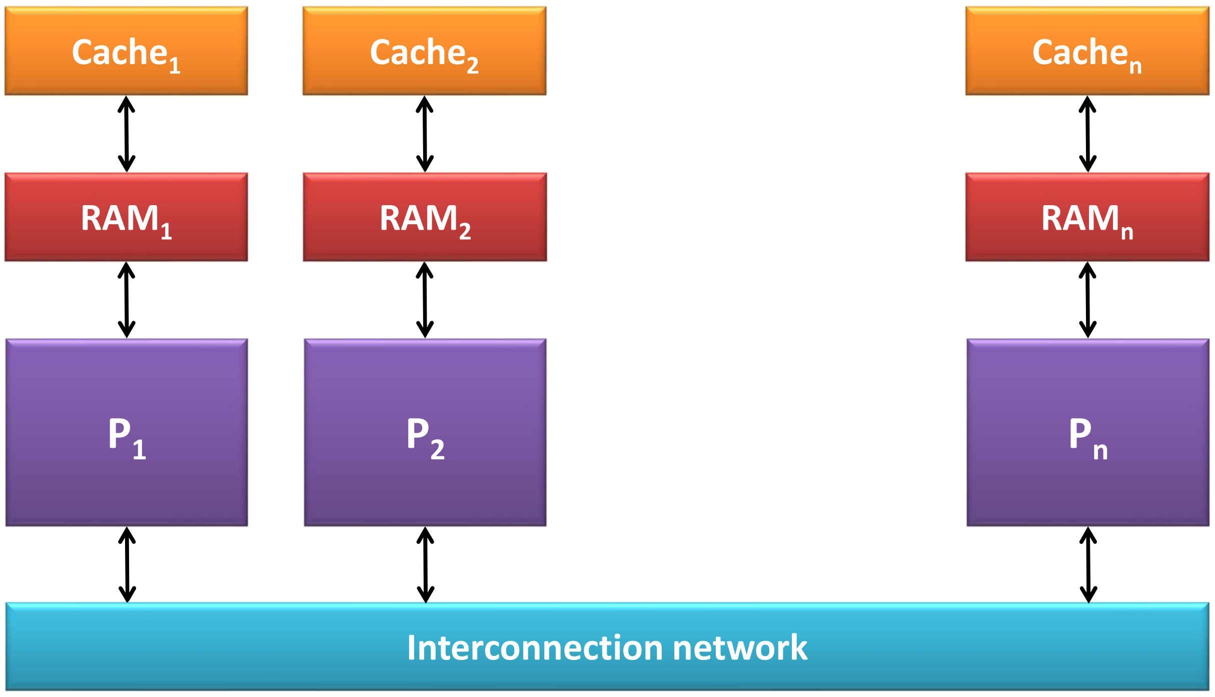


Figure 1.3: Non-Uniform Memory Access system.

At the software level, a multicore architecture will be useful when the code can be parallelized, that is, executed in different threads. As mentioned at the beginning, the software must be implemented specifically to perform this parallelization, which is challenging because it requires complex coordination of threads and control of access to shared resources. Also, debugging such programs is far from trivial since data is shared among the different threads. On the other hand, the performance of parallel applications in a multicore environment is limited by their non-parallel part that form bottlenecks. Therefore, for an optimal use of multicore processors, the non-parallel part has to be optimized, either by parallelizing the non-parallel part or by making it faster using more efficient algorithms [65].

When it comes to measuring the benefits of using this architecture, the most intuitive metric is the execution time, which is the total time the program takes from start to finish. However, there are other interesting metrics that can be useful depending on the objectives to be achieved. For instance, throughput, which is the average rate of how many processes have been successfully executed. There is also the so-called response time, which is the elapsed time between the request and the time when the system starts working on this request. The memory bandwidth measures the amount of data that can be moved simultaneously between the cores and the memory. With the current concern for the environment and global warming, the energy consumption is also a very important metric. These metrics can be categorized into three groups: higher is better, lower is better, and nominal is better. As an example, higher performance is better, shorter run time is better, and nominal energy consumption is better [65].

There are a large number of tools that can be used to program in these architectures, the following being some of the most well-known and widely used [9]:

- **OpenMP**: Open Multi-Processing is a programming interface that allows you to add concurrency to code written in C, C++, and Fortran. It is cross-platform, and is composed of a set of compiler directives, library routines, and global variables that affect the program at runtime. It is based on the fork-join model, which consists of dividing a heavy task into threads and then collecting the results of each of the threads to join them into a single result [7].

- **MPI**: Message Passing Interface is a communication protocol designed for use in programs that use multiple processors. It defines the syntax and semantics of a library of functions for message passing in multiprocessor environments. Some of its most important characteristics are that it is portable, and that it does not need shared memory, being therefore especially useful for synchronizing processes in distributed systems [17].
- **PThreads**: The POSIX Threads library is a library oriented to work with several threads simultaneously. It contains a set of interfaces for programming using threads that in terms of memory share the same data and heap segments, but each using its own stack. Since it complies with POSIX standards, any program made with pthreads will be portable to any POSIX operating system that supports threads [40].
- **Java threads**: Java threads have a similar purpose to pthreads, but unlike pthreads, Java threads are structured as objects.Each thread is an instance of the Thread class, and each time the object is created a separate task is created. However, Java also gives the possibility to use interfaces (provided by Java) to handle the tasks by passing them to an executor [43].
- **TBB**: The Threading Building Blocks library was developed by Intel to simplify the development of thread-based applications while offering scalability and efficiency. It is designed in C++, and includes options to automatically parallelize the routines chosen in user programs. However, its strong point is that it offers the programmer the possibility to control in detail the parallelism design [57].

GPU computing

Graphics Processing Units (GPUs) offer another HPC alternative for parallelizing and accelerating applications. A GPU is made up of hundreds or thousands of cores, with these cores organized into several multiprocessor arrays. Thus, any routine designed to run on a GPU (commonly called kernel), is divided into threads which in turn are grouped into blocks whose configuration must be set by the programmer depending on the problem and the GPU itself [37]. GPUs should be viewed as devices designed to act as co-processors to CPUs, and should never be viewed as separate processors [66]. A GPU always involves CPU-GPU interaction, a host/ device programming model where the CPU is the host and tells its GPU device what to do. Once the CPU tells the GPU the appropriate instruction, all it has to do is wait for the GPU to return the result of its work.

This CPU-GPU interaction has its advantages and disadvantages. The most obvious advantage is that a GPU greatly expands the computational potential of the system. The most important disadvantage comes from the fact that this interaction involves passing data from one device to another. The GPU's high operational capacity is justified when a significant amount of data is to be worked with, so transferring that data from the CPU to the GPU (and subsequently retrieving the results, i.e. a GPU-CPU transfer) is assumed to be a costly task in terms of time [37]. In an ideal case, it will only be necessary to transfer twice, at the beginning and at the end. In real applications, several transfers will be necessary, which could reduce the time gain achieved by the high computational speed of the GPU. This is because, despite the increasing use of GPUs

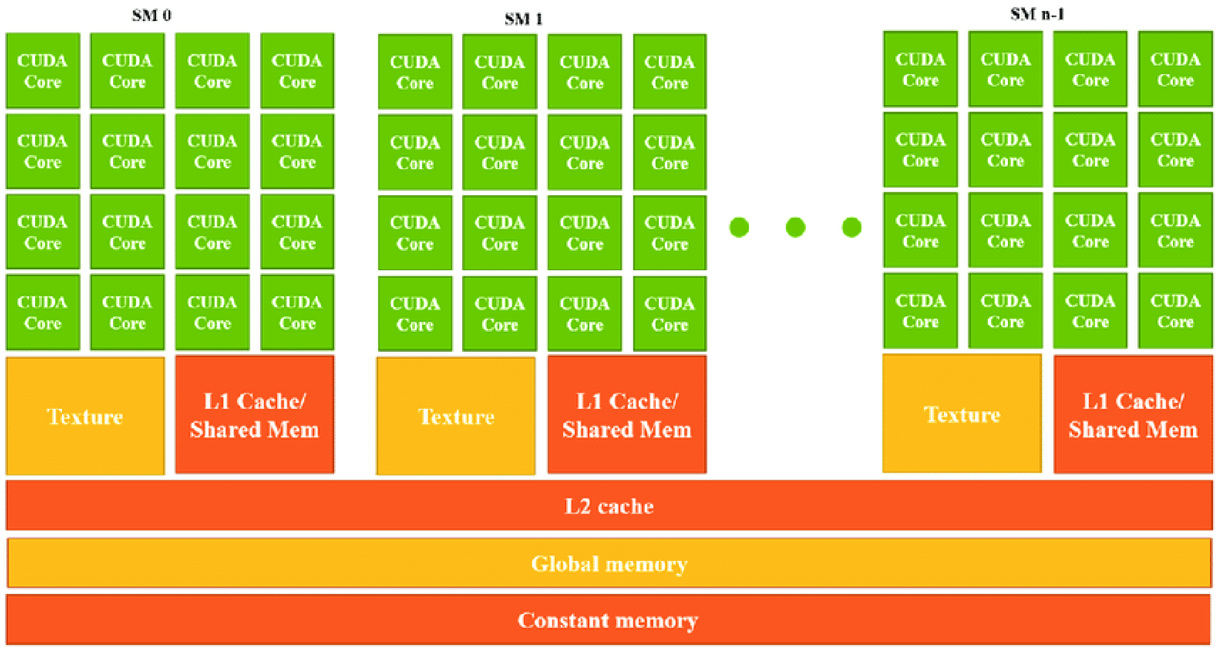


Figure 1.4: Example of GPU architecture.

to solve new problems, it is not a method that can be used to solve every problem, even if it involves a large amount of computation.

It has been mentioned that a GPU is composed of several processing units commonly called multiprocessors. In turn, each multiprocessor is composed of many cores. The number of multiprocessors and cores per multiprocessor depends on each particular GPU. Also, different GPU models have different memory architectures, but we can simplify by saying that as a rule, the cores of each multiprocessor share one memory, as well as there is a global memory shared among all multiprocessors. As is intuitive to understand, a core will access its multiprocessor's memory faster than the overall memory. Of course, proper use of these memories is critical to optimize the computer performance and should be a fundamental part of problem solving problems. As an example, Figure 1.4 shows an NVIDIA GPU architecture with n multiprocessors (labelled as SM). Each multiprocessor has 32 cores and its own cache memory. There are other 3 levels of memory, labelled as L2 cache, global memory, and constant memory, respectively. Each memory has its own characteristics.

The approach to parallelizing a task is different from that which would be used in a multicore system as described in the previous subsection, since millions and millions of threads can be launched on a GPU. Despite this large number of threads, it is essential to parallelize the problem properly so that it can be mapped to the GPU optimally. As a rule, there is a minimum number of threads (this unity is commonly called warp) that will be running at any given time. This is why the GPU executes programs based on warps. However, it is often the case that a warp is too small to be used as a unit. That is why kernels are launched in terms of blocks, which are simply a set of warps. At the programming level, problems arise in terms of threads/blocks. There is no block size that we can set as good, but this value will depend on the problem and, of course, on the GPU architecture [45].

The two most popular tools for GPU programming are Open Computing Language (OpenCL) and Compute Unified Device Architecture (CUDA). OpenCL is a programming language used to parallelize applications on both CPUs and GPUs. It enables a high-level abstraction for low-level hardware routines, as well as a consistent memory and execution model for parallel code execution [74]. However, CUDA is more widely used and it is the tool we use in the thesis. CUDA is a parallel program development platform that uses a variation of the C language for programming (although through wrappers it can also be used with Python, Fortran and Java). It was developed by NVIDIA in 2007, and although it has not stopped evolving since then, it only works on NVIDIA GPUs [15]. CUDA is based on the single instruction, multiple threads model, which implies that each instruction in the program is executed by hundreds of threads with different data. The mapping of threads to the GPU cores is performed automatically by CUDA [16].

1.2.2 Quantum computing

In quantum computing, the qubit is the basic unit of information. Like the bit, a qubit can be in one of two values. However, the most characteristic properties of the qubit are that (i - superposition) the qubit can be in both states at the same time, and that (ii - wave function collapse) only one of such values will be obtained if the qubit is measured. The state of a qubit can be expressed algebraically as follows:

$$c_1 |b_1\rangle + c_2 |b_2\rangle \tag{1.1}$$

This equation defines any quantum state, that is, the value of a qubit at a given time. In this equation, $|b_1\rangle$ and $|b_2\rangle$ (represented in Dirac notation) are the so-called base states, which for the moment can be interpreted as the values 0 and 1 used by the classical bit. On the other hand, c_1 and c_2 represent what percentage of the qubit is in $|b_1\rangle$ and $|b_2\rangle$ states, respectively. Suppose an experiment has n possible outcomes E_1, E_2,..., E_n. Associated with each output E_i is a probability p_i. The probabilities must be between 0 and 1, and must sum to 1. In the case of the qubit, the possible outputs are $|b_1\rangle$ and $|b_2\rangle$ with probabilities $p_1 = {c_1}^2$ and $p_2 = {c_2}^2$, respectively.

To accurately define a qubit and its associated state, complex numbers must be used. In particular, a two-dimensional complex space $\mathbb{C}^2$ is sufficient. However, for the purpose of exposing its operation, it is sufficient to use $\mathbb{R}^2$ [3]. In fact, it is only necessary to consider the unit vectors of $\mathbb{R}^2$, that is, the kets of the form:

$$|v\rangle = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, with\ {c_1}^2 + {c_2}^2 = 1 \tag{1.2}$$

On the other hand, b_1 and b_2 can be any two vectors that form an ordered orthonormal basis $(|b_1\rangle, |b_2\rangle)$.

In quantum computing, the action of measuring affects the result. In quantum mechanics, we work with small particles such as atoms and electrons. To measure, we need to interact with them, so we affect them. Suppose a qubit q is in the state:

$$|q\rangle = 1\begin{bmatrix}1\\0\end{bmatrix} + 0\begin{bmatrix}0\\1\end{bmatrix} \tag{1.3}$$

For the sake of readability, we will also label the following quantum states:

$$\begin{aligned}
|0\rangle &= \begin{bmatrix}1\\0\end{bmatrix}\\
|1\rangle &= \begin{bmatrix}0\\1\end{bmatrix}\\
|+\rangle &= \begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}\\
|-\rangle &= \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}
\end{aligned}$$

Thus, equation 1.3 will be redefined as:

$$|q\rangle = 1\,|0\rangle + 0\,|1\rangle \tag{1.4}$$

Probability says that measuring q will always result in $|0\rangle$ if we measure with the bases $(|0\rangle, |1\rangle)$. If we measure the same qubit n consecutive times in this way, we will get the same result every time. However, if we use different bases $(|+\rangle, |-\rangle)$, we will surprisingly get another result [3]. Although the idea itself is counter-intuitive, it can be easily demonstrated mathematically by simply expressing q as a linear combination of the new basis vectors:

$$\begin{aligned}
\begin{bmatrix}1\\0\end{bmatrix} &= c_1\begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix} + c_2\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}\\
A = (\begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}, \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}) &= \begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}\\
A^T\begin{bmatrix}1\\0\end{bmatrix} &= \begin{bmatrix}\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}\\
|q\rangle &= \frac{1}{\sqrt{2}}\begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}
\end{aligned} \tag{1.5}$$

We say that two states are equivalent if they give the same result when they are measured -their priorities are similar-. However, and based on what has just been mentioned about bases, some states could be equivalent if they measured in a certain direction, but not be equivalent if measured in another direction. For example, the following two states are equivalent if we

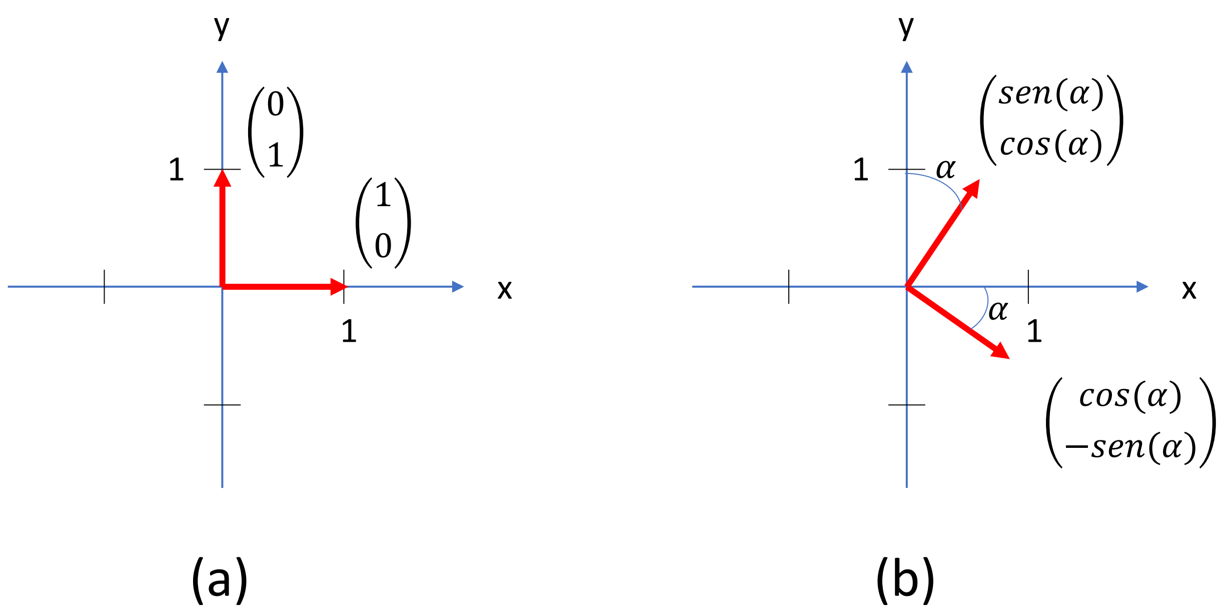

y
1
$\begin{pmatrix}0\\1\end{pmatrix}$
$\begin{pmatrix}1\\0\end{pmatrix}$
x
1
(a)
y
1
α
$\begin{pmatrix}sen(\alpha)\\cos(\alpha)\end{pmatrix}$
x
α
1
$\begin{pmatrix}cos(\alpha)\\-sen(\alpha)\end{pmatrix}$
(b)

Figure 1.5: (a) Standard Basis $(|0\rangle, |1\rangle)$. (b) $(|0\rangle, |1\rangle)$ rotated $\alpha°$.

measure on the basis $(|0\rangle, |1\rangle)$, but they are not if we measure on the basis $(|+\rangle, |-\rangle)$. This is because, beyond the probability of the outcome, the state is still important for the next operations to be performed on the qubit. These operations can be translated into moving the qubit by a certain angle.

Continuing with our 2D simplification, we could represent the basis $(|0\rangle, |1\rangle)$ as shown in Figure 1.5(a). Rotating this base by an angle $\alpha°$ would be equivalent to performing the operation shown in Figure 1.5(b). If we rotate an angle of 90 degrees we arrive at a base equivalent to the original but exchanging the elements. Or said correctly, exchanging the probabilities of each base:

$$\left(\begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right) \tag{1.6}$$

In pure measurement terms, we denote θ as the angle by which we rotate the measuring apparatus with which we measure the qubit, while α is the angle by which we rotate the basis vectors. A complete set of directions ranges from $0°$ to $180°$, while a complete set of rotated bases ranges from $0°$ to $90°$. If we reach $\theta = 180°$ or $\alpha = 90°$, the two elements of the original basis are exchanged. Thus, we can establish that:

$$\theta = 2\alpha \tag{1.7}$$

Therefore, the basis associated with the rotation of an angle θ of the measuring device will be given by:

$$(\begin{bmatrix} cos(\frac{\theta}{2}) \\ -sen(\frac{\theta}{2}) \end{bmatrix}, \begin{bmatrix} sen(\frac{\theta}{2}) \\ cos(\frac{\theta}{2}) \end{bmatrix}) \tag{1.8}$$

This defines the relationship between the angle of rotation of the bases on which it measures, with the angle of rotation of the measuring device itself. We have demonstrated the importance of these rotations in the result, but we have yet to expose the rotations that the qubits can undergo, which are the ones that will really allow us to operate in a quantum computer.

Quantum gates

For simplicity, in the following it will be understood that we are working with the basis $(|0\rangle, |1\rangle)$. As recently described, we can express any qubit in the form:

$$|q\rangle = c_1 |0\rangle + c_2 |1\rangle \tag{1.9}$$

Also for simplicity, we will assume that our measuring devices will be fixed and will not rotate. Thus, we can define a quantum gate as a quantum version of the logic gates used in quantum circuits. Mathematically speaking, quantum gates are orthogonal arrays that allow us to operate (rotate) on the qubit. In digital electronics, there are only two gates that operate on a single bit: the identity gate and the NOT gate. In quantum computing, since the only requirement is that the matrix representing the gate (which, for obvious algebraic reasons, must have the size 2×2) must be orthogonal, there are infinite possibilities.

Among the infinite quantum gates that exist, there are four particularly well-known and widely used ones: the so-called Pauli gates. The first gate, the gate I, consists of the matrix identity and therefore leaves the qubit in the state it was originally in without producing any change in it. The other three Pauli gates, called X, Y and Z, do produce interesting changes in the qubit. The X and Y gates exchange the probabilistic amplitudes of $|0\rangle$ and $|1\rangle$ with each other. The Y gate also changes the relative phase of the qubit. The Z gate leaves the probabilistic amplitude of $|0\rangle$ intact, but changes the sign of the probabilistic amplitude of $|1\rangle$. The effect of each of these gates on a qubit in its general state is shown in Figure 1.6.

Two other key gates are the Hadamard gate and the CNOT gate. These gates will introduce the concepts of superposition and entanglement to quantum circuits. We must also highlight the T gates for their usefulness in fault tolerance in quantum computing. All the circuits realized as part of this thesis can be built using the set of quantum gates presented in this section.

Quantum search algorithms

Several algorithms have already demonstrated the superiority of quantum computing over classical computing. Among these algorithms, Grover's algorithm and its many variants constitute a particularly effective class of algorithms. The problem that this type of algorithm solves is defined as follows: given a search space of size N, and without prior knowledge of how the

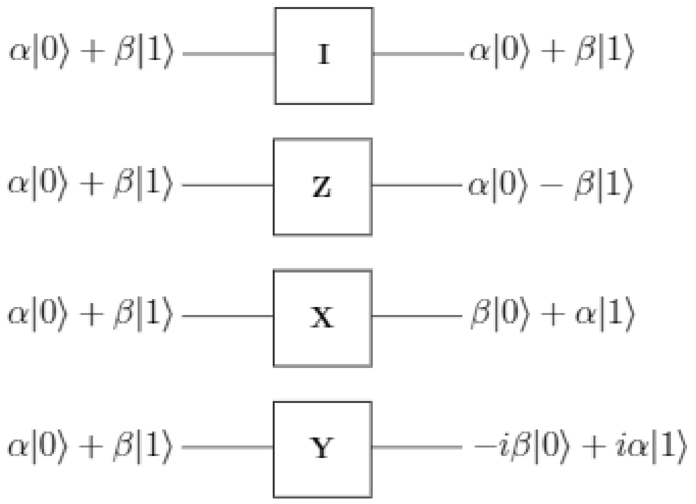

$\alpha|0\rangle + \beta|1\rangle$
I
$\alpha|0\rangle + \beta|1\rangle$
$\alpha|0\rangle + \beta|1\rangle$
Z
$\alpha|0\rangle - \beta|1\rangle$
$\alpha|0\rangle + \beta|1\rangle$
X
$\beta|0\rangle + \alpha|1\rangle$
$\alpha|0\rangle + \beta|1\rangle$
Y
$-i\beta|0\rangle + i\alpha|1\rangle$

Figure 1.6: Effect of Pauli gates on a qubit. Image extracted from [1].

information is structured, we want to find an element that satisfies a certain condition. Using classical computation, solving the problem requires approximately N operations. However, a quantum computer can solve it in approximately $\sqrt{N}$ operations [41].

The operation of quantum search algorithms is shown in Figure 1.7. The core of it is the small subcircuit called Oracle. The mission of this subcircuit is to check whether or not an element x satisfies the search condition, so it must return 1 if x satisfies the condition, and 0 if it does not. Of course, the oracle/circuit depends entirely on the specific problem to be solved and is a part of the algorithm that must be customized to suit each specific problem. Precisely, the difficulty of applying the algorithm to a problem (assuming that the problem is tractable with this algorithm), is to build an efficient oracle. Mathematically, we can define the oracle O as a circuit that performs the following action:

$$|x\rangle\,|q\rangle \to |x\rangle\,|q \oplus f(x)\rangle \tag{1.10}$$

where $f(x)$ is the function that determines whether or not x satisfies the condition, and q is a qubit that is flipped if and only if x satisfies the condition.

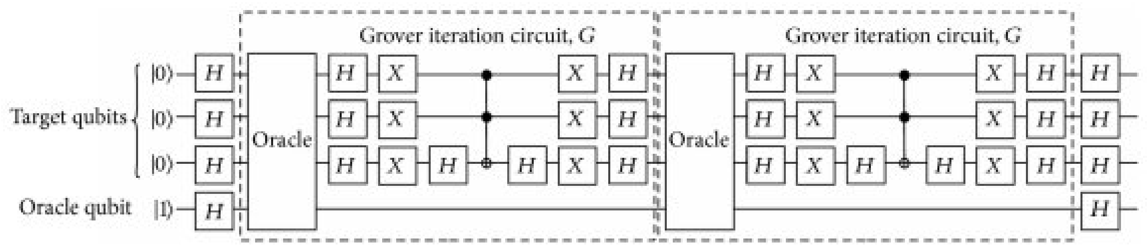


Figure 1.7: Schematic circuit for Grover's algorithm. Image extracted from [28].

The rest of the algorithm is always the same (Figure 1.7), regardless of the problem being solved. The gate denoted with H is the Hadamard gate discussed in the previous subsection, which is responsible for putting the qubits in superposition. Subsequently, the G circuit is applied, which consists of 4 sequential steps: the oracle, applying the Hadamard gates, applying a conditional phase shift (with the Pauli X gates) and, again, applying the Hadamard gates. What the G circuit essentially does is increase the probabilistic amplitude of the result. G must be repeated the appropriate number of times so that the probability of the desired outcome is sufficiently high. Identifying the number of times to apply G depends on the problem, but as an illustration, we can say that in an N-element problem with M solutions, it is necessary to apply it $O(\sqrt{N/M})$ times to find a solution.

1.3 Thesis contributions in the design of quantum circuits

A quantum computer is programmed by circuits. Similar to the way a program is made in a classical computer, every quantum circuit has a specific design for the problem it must solve. An example already discussed earlier is that of Grover's algorithm oracle, which must have a design dependent on the problem being addressed. Small and optimized quantum circuits are a very valuable asset even when they do not offer any quantum advantage, since they can be used as part of larger circuits to implement quantum algorithms [56]. An important part of the present thesis has been, precisely, to design and implement optimized quantum circuits that would allow progress in the construction of quantum algorithms to solve the problems addressed in the thesis (discussed in the next section). To this end, our initial efforts were focused on an intensive study of the state of the art of quantum circuits for certain arithmetic operations.

The first problem we encountered was the lack of standard metrics to define the goodness of such circuits. That is why we focused on establishing a measurement framework that would provide as much information as possible about each circuit so that any interested researcher could use our work to quickly and visually obtain this information and choose the right circuit for their needs. This framework is based (but not limited to) on the work developed by Mohammadi et al [36]. Our contribution has been to search and measure the circuits in the literature using the same metrics to catalogue and coherently compare them, making this information public domain. The most relevant metrics included in such a framework are:

- Quantum cost: the quantum cost of quantum gates that act over one or two qubits is defined as 1. The quantum cost of any other gate will be the sum of the quantum costs of the gates that compose it.
- Delay: the delay of a quantum gate that acts over one or two qubits is set to $1\triangle$. The delay of a circuit consists of the sum of the delay of the gates it has in its critical path.
- Ancilla qubits: qubits that are used to perform auxiliary operations, but not to contain an input of the circuit.
- Garbage outputs: qubits that are not part of the solution given by a circuit and that contain unknown values. The value of these qubits must be restored (uncomputed) to their initial value or these qubits could not be entangled with qubits of other circuits.
- T-count: the number of T gates a circuit has. This gate is expensive in comparison to the rest of the gates, so this metric can give more valuable information than the quantum cost in certain cases.

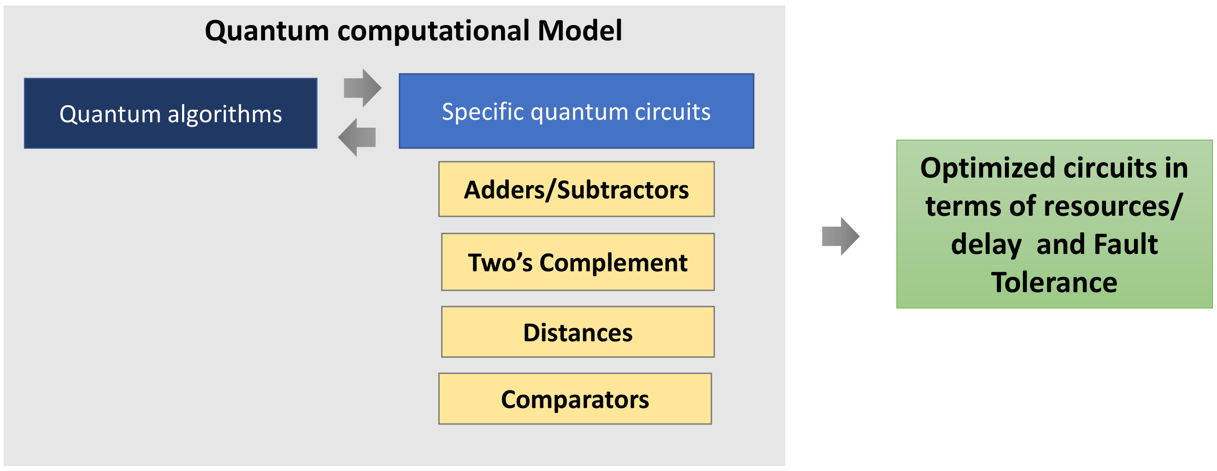

Quantum computational Model
Quantum algorithms
Specific quantum circuits
Adders/Subtractors
Two's Complement
Distances
Comparators
Optimized circuits in terms of resources/ delay and Fault Tolerance

Figure 1.8: We have studied and designed optimized quantum circuits in several fields to be able to perform major operations using quantum computing.

- T-depth: the number of T gates a circuit has in its critical path. This gate is very slow in comparison to the rest of the gates, so this metric can give more valuable information than the delay in certain cases.

There are a large number of existing circuits in the literature and cataloguing them has been a productive work. However, sometimes even the best circuit of its type is not optimized enough to allow certain algorithms to be performed on a particular machine due to its limited resources. That is why we have also designed and produced optimized circuits to advance in these fields. Figure 1.22 shows a schematic of the operations we have addressed. We discuss these areas and our progress in each of them below.

Adders/Subtractors

Adders are possibly the most important circuit in quantum computing today. This is because addition is a fundamental operation in Shor's algorithm. In fact, it is the main bottleneck in the implementation of this algorithm, so there is a major research race to obtain the best possible adders. As a result, there is a wide variety of adder circuits, which includes circuits of various types and methodologies.

The contribution of the thesis in the field of adders comes from a thorough review of the circuits of this category available in the literature [48]. Some of these circuits also perform the subtraction operation (these circuits are called adder/subtractors), so the study of this type of circuits has also been included. As part of this review, more than 40 references have been searched, measured, and classified, summarizing all this information in several detailed tables for quick reference (Tables 1.1, 1.2, 1.3, and 1.4).

Adder	Quantum cost	Delay	Auxiliary inputs	Garbage outputs	Adder/ subtractor
[25]	14	$14\triangle$	3	3	Yes
[41]	6	$6\triangle$	1	0	
[81]	5	$5\triangle$	1	0	
[62]	5	$5\triangle$	1	0	Yes
[22]	4	$4\triangle$	1	0	

Table 1.1: Comparison between half adders.

Adder	Quantum cost	Delay	Auxiliary inputs	Garbage outputs	Adder/ subtractor	Fault tolerant
[61]	18	$18\triangle$	5	6	Yes	
[81]	15	$15\triangle$	3	0		
[62]	15	$10\triangle$	1	0	Yes	
[10]	14	$14\triangle$	0	0		
[22]	12	$12\triangle$	3	0		
[38]	12	$12\triangle$	1	0		
[35]	11	$11\triangle$	2	3		Yes
[26]	11	$11\triangle$	4	4	Yes	
[76]	10	$8\triangle$	1	0		
[59]	10	$10\triangle$	1	3	Yes	
[64]	8	$8\triangle$	1	1		
[4]	8	$8\triangle$	1	1		
[69][b]	8	$5\triangle$	2	0	Yes	
[84]	8	$7\triangle$	2	2		Yes
[32]	6	$4\triangle$	1	1		
[69][a]	6	$4\triangle$	1	0		

Table 1.2: Comparison between full adders.

Adder	Quantum cost	Delay	Auxiliary inputs	Garbage outputs	C_{in}
[67]	$26N-29$	$24N-27\triangle$	0	0	
[10]	$17N-12$	$10N\triangle$	1	0	
[68]	$15N-9$	$13N-7\triangle$	0	0	
[72]	$15N-6$	$9N+5\triangle$	0	0	Yes
[73]	$13N-8$	$11N-4\triangle$	0	0	
[38]	$12N$	$10N\triangle$	$4N$	0	Yes

Table 1.3: Comparison between ripple-carry adders.

Adder	Quantum cost	Delay
[13]	$28N - 15W(N) - 15log(N) - 6$	$logN + logN/3 + 7\triangle$
[70]	$26N - 15W(N) - 15log(N-4)$	$logN + logN/3 + 2\triangle$

Table 1.4: Comparison between carry-lookahead adders. $W(N)$ is the number of ones in the binary expansion of N.

On the other hand, we have also made a contribution in the form of three fault-tolerant optimized adder circuits. These circuits are each focused on reducing the T-cost, the T-depth, and/or the number of qubits needed to perform the summation in a noise mitigation context. The design of these adders has been achieved by combining computer structure techniques with the use of a special gate that allows the use of T-gates to be reduced without compromising the reliability of the result obtained with it. The metrics of each of the three proposed adders are shown in Table 1.3. More details about the review can be found in our paper [48].

Adder	**T-depth**	**T-count**	**Ancilla qubits**
(Out-FT-QCLA1) [71]	10	36	10
(Out-FT-QCLA2) [71]	13	48	6
(In-FT-QCLA1) [71]	20	56	13
(In-FT-QCLA2) [71]	22	76	5
New Adder 1	10	36	9
New Adder 2	14	51	5
New Adder 3	12	41	6

Table 1.5: Depth and cost comparisons between the most optimized 4-digit adders in terms of T gates.

Two's complement

The two's complement operation is fundamental in classical computing. We wanted to study its possible advantages in quantum computing. This operation allows, among other things, to perform subtractions employing adders, which together with the large availability of adders (as explained in the previous subsection) allows to increase and optimize the possibilities to perform subtraction in a quantum computer, both in terms of quantum cost and speed.

Our contribution in this operation consists in the design of four binary sign/magnitude to two's complement converters [51, 52]. These converters will allow working with two's complement in quantum computers, each of them offering its own advantages. The first converter was designed to be used in a free-noise environment (i.e. quantum simulators). Therefore, it is non-fault tolerant, but it is optimized in terms of delay and is the fastest circuit of its kind. Its design is based on the propagation of classical carry-lookahead adders, but adapting it to a quantum environment while maintaining reversibility and in a context free of garbage outputs. The circuit is shown in Fig. 1.9.

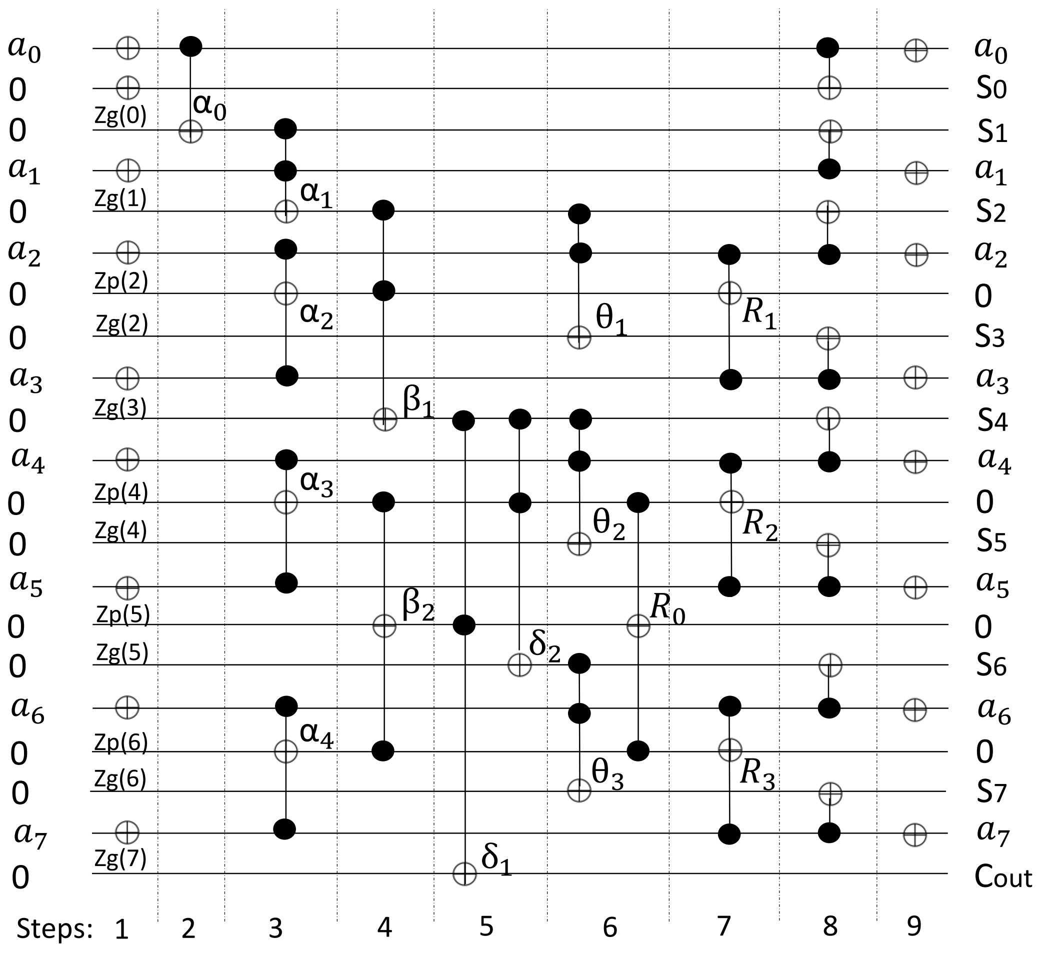
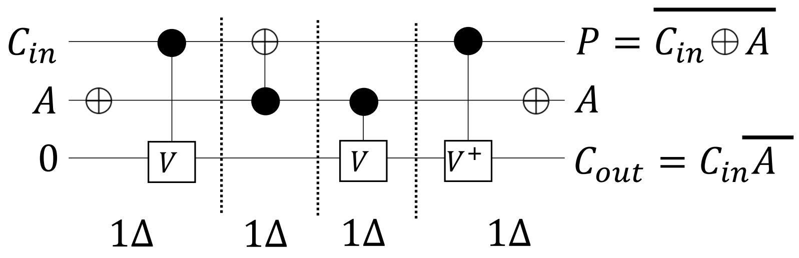


Figure 1.9: First proposed converted. a_i are the qubits of the input number, $Zg(i)$ will contain S_{i+1} (the result) at the end, and $Zp(i)$ are auxiliary qubits used to store propagated carry value for intermediate digits (they are uncomputed at the end).



C_{in}
A
0
V
V
V^+
$P = \overline{C_{in} \oplus A}$
A
$C_{out} = C_{in}\overline{A}$
1Δ
1Δ
1Δ
1Δ

Figure 1.10: Second proposed converter. C_{in} is the input carry, A is the input, P is the output and C_{out} is the output carry. It has 1 auxiliary input and no garbage outputs. Its quantum cost is 6 and it has a delay of 4$\triangle$.

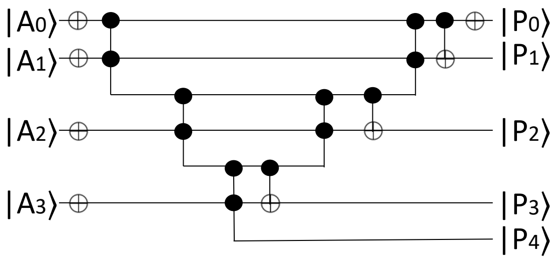


Figure 1.11: Fourth proposed converter, for the 4-digits case.

Based on the idea of this first converter, we developed a second circuit but this time based on the operation of ripple-carry adders. As a result, this second design consists of a converter that, digit by digit, performs the conversion sequentially so that its quantum cost is the most optimized of the circuits in the literature of this type. The circuit is shown in Fig. 1.10.

For the design of the other two circuits we focused on making them fault tolerant. In particular, the third circuit is an almost direct conversion of the previous one (the second one) to a fault-tolerant environment, trying to optimize this conversion as much as possible so that neither the quantum cost nor the T-cost would increase. However, in this aspect our greatest achievement came with the fourth circuit. Instead of starting from a previous design, the fourth circuit has a design of its own. In combination with the selection of the appropriate quantum gates (Figure 1.11), the fourth converter improves the existing converters in terms of fault tolerance and T-gate related measurements. These two circuits are also free of garbage outputs.

Comparators

A comparator circuit compares two numbers A and B, and returns a result that indicates if $A > B$, $A = B$, or $A < B$. In classical computing, this kind of circuit usually returns -1, 0, and 1, respectively. There is another kind of comparator called half-comparator, which is able to compute if $a < B$, or if $A \geq B$. These circuits are widely used in digital electronics, in operations as common as, for instance, checking that a value does not exceed a certain threshold.

We developed two quantum comparators focused on be fault-tolerant circuits in order to reduce the effects of the internal and external noise [49]. Since the objetive here was to design valid comparators for real quantum computation, the main metrics to optimize were the T-count

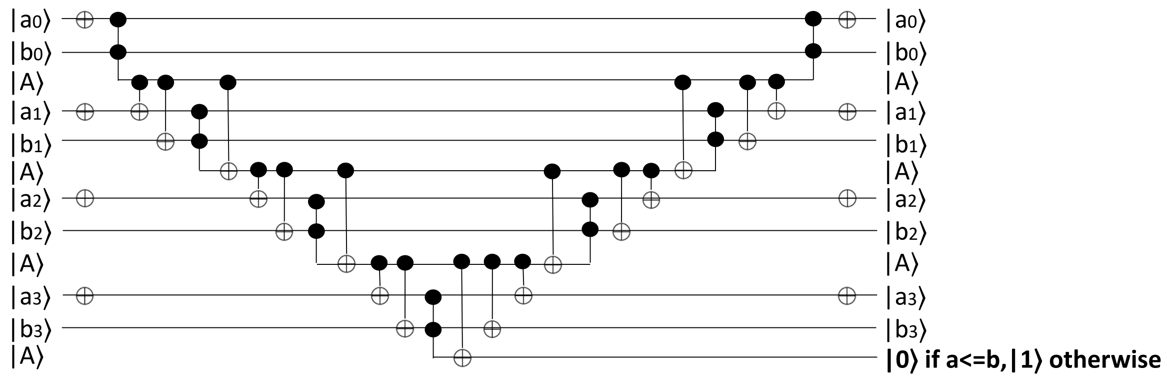


Figure 1.12: First proposed comparator for the $N = 4$ case. This circuit is focused on reducing the T-count. a_i and b_i are the bit strings to be compared. A are ancilla qubits.

and the T-depth. Moreover, a circuit with a small quantum cost will be less affected by noise than a larger one, since each quantum gate used has its own probability of failure due to noise. On the other hand, qubits slowly lose their value, so a circuit that is faster than another (that is, with lower delay) will be less prone to failure than a faster one. These parameters must therefore also be taken into account when designing.

The first comparator was focused on this first metric. It is based on the methodology used on the most optimized adder in these terms, but working in two's complement and saving several operations to compute the half-comparison between the two inputs. A fundamental part in achieving the reduction in the number of T gates was the inclusion of the so-called temporary logical-AND gate. This gate performs a similar operation to the Toffoli gate, but with a considerable saving in the T-count, especially in its uncomputation. Thanks to the original methodology, the techniques employed, and the use of this gate, the proposed circuit is the best comparator in terms of T-count. An example for the case $N = 4$ is shown in Fig. 1.12.

In turn, a second comparator was designed, but this time focused on the T-depth and the delay. This circuit is $O(Lg(N)$, and it was designed based on the methodology of another efficient adder. It is no surprise that the adder on which it is based is the most optimized in terms of T-depth. The fast calculation of the carry-over, which in the case of the comparison is the result of the circuit, allows the operation to be completed in the above-mentioned logarithmic time. Uncomputing the rest of the circuit adds extra time to the critical path, but it is worth it in the interest of saving a large number of garbage outputs. Fig. 1.13 shows an example of the second comparator, for the $N = 8$ case.

Distances

The calculation of distances between two points is one of the most common operations in classical computing. There are several types of distances, such as Geodesic and Euclidean ones. Focusing on the latter, the distance between two points $A(x_1, y_1, z_1)$ and $B(x_2, y_2, z_2)$ would be defined by the formula $d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$. In certain contexts, for example in the context of comparing distances, you can work with the square of these distances. In this way, the square root operation of the above formula is not necessary. It may seem like a

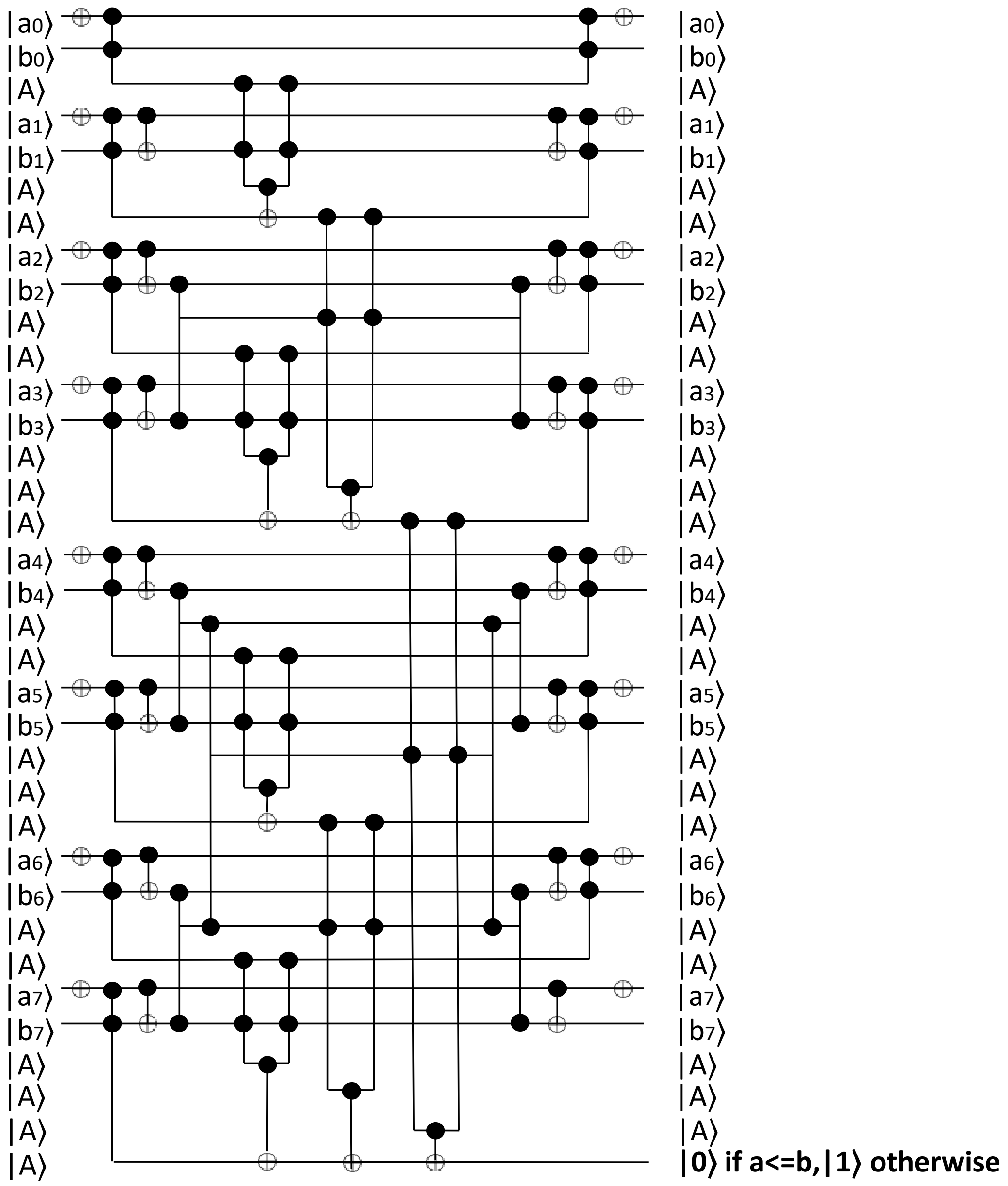


Figure 1.13: Second proposed comparator for the $N = 8$ case. This circuit is focused on reducing the T-depth. a_i and b_i are the bit strings to be compared. A are ancilla qubits.

negligible computational saving in classical computing, but it is nevertheless a significant saving in classical computing due to the complexity of performing square roots with qubits. Thus, the computation of the distance between two points can be calculated by performing the following operations:

1. To calculate $d_x = x_i - x_j$, $d_y = z_i - z_j$, $d_z = z_i - z_j$.
2. To calculate the squares ${d_x}^2$, ${d_y}^2$, ${d_z}^2$.
3. To compute the addition $d^2 = {d_x}^2 + {d_y}^2 + {d_z}^2$.

Building a circuit that computes d^2 will involve, if working with three dimensions, three subtractions, three squares, and two additions. The required number of qubits will depend on the size of the data we want to work with, but we can already deduce in advance that it will not be a small number even for a small prototype. In relation to the above, it is worth noting that the square will greatly increase the need for qubits, so it is not possible to stay at the initial size of the coordinates to estimate the required number of qubits. Similar to what has been done with the adders, a thorough review is necessary here to find the most suitable type of circuit to perform each of the three types of operations. The resulting circuit will be a clear example of the importance that we pointed out at the beginning of this section that small circuits are a very precious asset, because only with them we will be able to implement this type of larger circuits without exceeding the resources we have.

Fortunately, at the time of undertaking this circuit, we already had the adder review, which as we have said is the most abundant kind of arithmetic circuit. The same review process had to be repeated with circuits that allow us to calculate the square of a number, as well as with subtractors. Regarding the first type, it is true that there are not a large number of them in the literature. The second type (subtractors) is more abundant, although there are not as many as in the case of adders. Nevertheless, and as seen in the previous subsection, subtraction can be performed by means of a two's complement adder, so this review should include a comparison with all the adders adapted to work with two's complement. For this assumption, we differentiate two cases: the possibility of using converters, or the possibility of adapting the adders by means of certain techniques so that they can work directly in two's complement (something that the adders that support carry can do in a simple and natural way).

The prototype built, despite admitting a reduced size of coordinates and being limited to a maximum of three x,y,z coordinates, has been a success and is in the process to be published in an international journal at the time of writing these lines. The justification of this circuit in quantum computing is explained in the next section.

1.4 Applications

The current subsection provides a formal definition of the kind of problems addressed as part of this thesis and the main strategies used to solve them. While the quantum computing paradigm used in the thesis was explained in Subsection 1.2.2, the applications described in this section have been achieved by developing the circuits presented in Subsection 1.3.

Fig. 1.14 summarizes the three applications -Microrheology, dimensionality redution using MultiDimensional Scaling, and Image Processing- and shows a simple definition of the involved techniques. It is relevant to mention that the term "classical computing" is used to differentiate any non-quantum computing from quantum computing. In particular, the "classical computing", which was described in Subsection 1.2.1, involves high performance techniques, such as multicore and GPU computing.

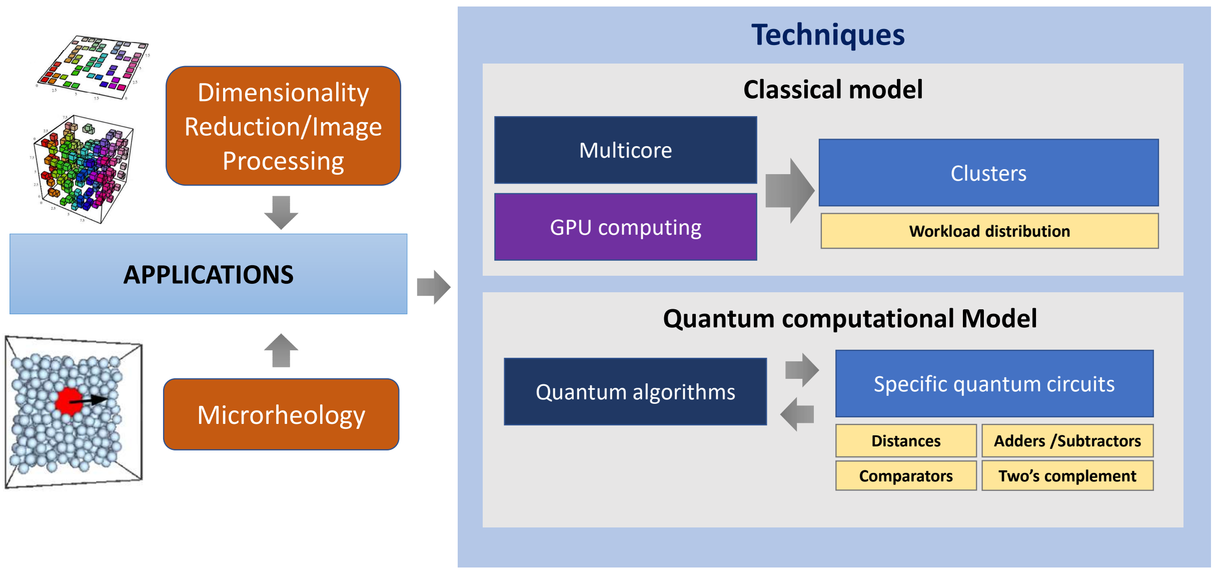


Figure 1.14: This thesis covers three problems: Microrheology, MultiDimensional Scaling, and Image Processing. Both classical and quantum computing techniques are used to address them.

1.4.1 Microrheology

Active microrheology is a technique to find the properties of viscoelastic materials. In particular, the use of this technique involves pulling a colloidal tracer into such materials, studying the dynamics, and interpreting the results. In this context, a material is usually represented as a bath of hard spheres, and the tracers have a size comparable to the bath particles, with Langevin, Stokesian, or Brownian dynamics [58]. In active microrheology, the most simple observable is the tracer mean velocity v, that can be used to calculate the effective friction coefficient, γ_{eff} of the bath via the stationary state relation $\mathbf{F}_{ext} = \gamma_{\text{eff}} v$, where $\mathbf{F}_{ext}$ is the force applied to the tracer. γ_{eff} is the microscopic equivalent to the viscosity, which characterizes the rheological behaviour of a given material (identifying Newtonian fluids, viscoelastic materials, ...). Therefore, the strategy is to measure experimentally, or simulate, the trajectory of the tracer subjected to an external force, to obtain the average velocity and thus the effective friction coefficient. The high computational cost of this strategy will be demonstrated below.

Our simulations of such materials have been done considering N Brownian particles in a cubic box [53, 54]. These particles include the tracer one, labelled with $j = 1$. Simulations in a system of quasi-hard Brownian spheres are strongly affected by finite size effects [20]. To palliate these effects and to obtain realistic measures, simulations of different sizes are required. The systems have been modeled using the Langevin equation of motion [12]:

$$m_j \frac{d^2 \mathbf{r}_j}{dt^2} = \sum_{i \neq j} \mathbf{F}_{ij} - \gamma_j \frac{d\mathbf{r}_j}{dt} + \mathbf{f}_j(t) + \mathbf{F}_{ext} \delta_{j1} \tag{1.11}$$

where:

- j is the number of the particle, with $0 < j <= N$.
- m_j is the particle mass.
- $\mathbf{F}_{ij}$ represents the interaction force between particles i and j.
- The friction force $\frac{d\mathbf{r}_j}{dt}$ is proportional to the velocity of the particle. The proportionality constant, γ_j, is calculated a $\gamma_j = \gamma_0 a_j$, where a_j is the radius of the particle.
- $f(t)$, the Brownian force, is random but related to the friction force.
- $\mathbf{F}_{ext}$, the external force, is caused by the tracer.
- δ_{j1} is the Dirac-delta symbol.

The interaction between two particles i and j can be computed from the central inverse-power potential as follows [27]:

$$V(\mathbf{r}) = k_B T \left(\frac{r}{a_{ij}} \right)^{-36} \tag{1.12}$$

being $r = |r|$ y a_{ij} the distance between such particles. All particles have the same value of m.

We have mentioned that finite-size effects must be considered since the simulations consist of a cubic array of tracers launched into an infinite bath [20]. Therefore, it is used a model based on solving the Navier-Stokes equation for an infinite array in a Newtonian fluid to analyze the results of the simulations [20]. In this model, γ_{eff} is related to the lattice spacing, L, as follows:

$$\frac{1}{\gamma_{\text{eff}}} = \frac{1}{\gamma_\infty} \left(1 - \frac{c}{L}\right) \tag{1.13}$$

being γ_∞ the coefficient in an infinite system, and c a constant related with the array structure and whose value in our case is $c = 2.8373a_t$ [20]. a_t is the tracer radius.

One approach to this problem involves computing simulations of systems of different sizes. This is necessary to extrapolate the friction coefficient γ_∞. Simulations of systems with sizes $N = 216$, 512, 1000, 2197, 4096, 8000, 15625 and 32768 particles has been carried out. In these simulations, energy is measured in units of the thermal energy $k_B T$, mass is measured in units of the particle mass m, and all lengths in units of the mean bath particle radius a. More details can be found in reference [12].

It is necessary to compute several tracer trajectories for various sizes to obtain an accurate value of the friction coefficient. In terms of computation, there are three parallelism levels in the model: 1) the computation of a single trajectory, 2) the computation of several trajectories, and 3) the simulation of sets of trajectories for several sizes to finally extrapolate γ_∞. For the sake of clarity, a schema of these levels is shown in Fig. 1.15. Moreover, this figure shows a graph where the abscissa axis shows the inverse system size, and the ordinate axis shows the

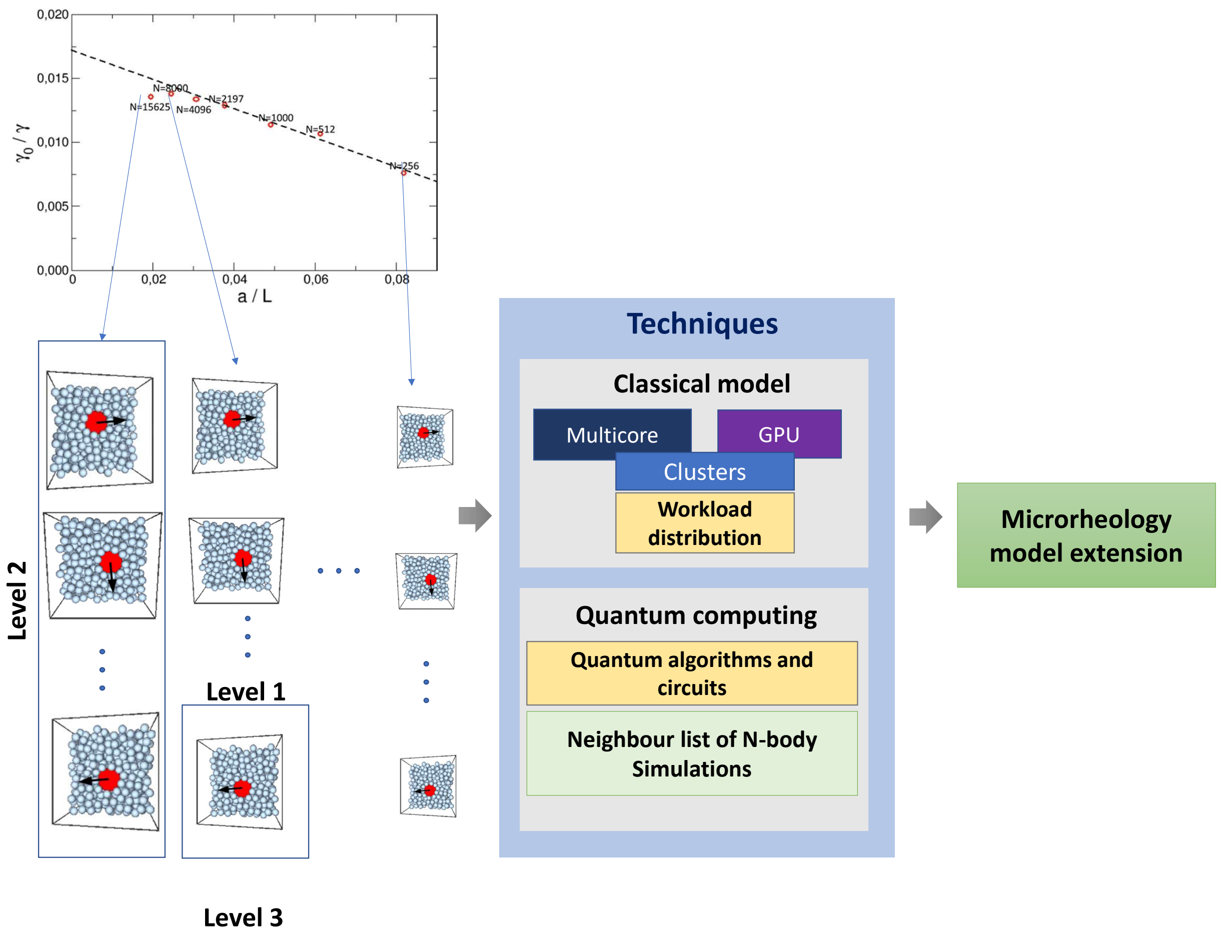


Figure 1.15: A microrheology simulation can be divided in three levels: 1) the computation of a trajectory, 2) the computation of all the needed trajectories for an specific size, and 3) the computation of the trajectories of each size.

normalized inverse effective friction coefficient. The aim is to find the values of the friction coefficient for very large N sizes and thus extend the microrheology models.

To address the parallelization of the microrheology model, different strategies have been considered on each level. Focusing now on the first level, a GPU code implemented in C and CUDA has been developed to accelerate the simulation of a single trajectory of the tracer particle. Algorithm 1 shows how a trajectory on a bath of N particles is computed. As can be seen, there is a GPU involved in the process. Each iteration of this algorithm not only evaluates the position and velocity of all the particles but also gets the list of neighbors of every particle. The complexity of these operations is $O(N)$ and $O(N^2)$, respectively.

On the one hand, the computation of the positions and velocities involves several steps. The initial values are transferred to the GPU. Then, the interaction forces among each pair of particles (F_{ij}) are computed. The positions and velocities are updated using the calculated forces, and F_{ij} is computed one more time to take into account the new positions. The velocities are again updated according to the equations of motion, and finally, the center of mass velocity is computed. This part is globally computed using 5 kernels (cuForces, cuUpPosVel1, cuUpPosVel2, cuVelVMC,

Algorithm 1 Host pseudocode of the model of bath of Brownian quasi-hard spheres.

```
Require:
    $ntraj$: the number of trajectories,
    $ttraj$: the total time steps of every trajectory,
    $\delta t$: time step,
    $N$: the number of particles,
    $\gamma_0$: solvent friction coefficient,
    $F_{\text{ext}}$: external force,
    $a[]$: vector to store the radius of the particles ($a[0] = a_t$ and $a[i] = a_b$),
    $\phi$: volume fraction,
    $Init\_Part[N]$: initial spatial locations and velocities of the $N$ particles,
    $mn$: the maximum number of neighbors of a particle,
    $\vec{Pos}[N]$: auxiliary structure to store the position of every particle.
    $\vec{Vel}[N]$: auxiliary structure to store the velocity of every particle.
    $NList[N][mn]$: auxiliary structure composed by a list which will store the neighbors of every particle.

Ensure:
    $\vec{T}[i][t]$: spatial location vector for the tracer at time $t$ in the trajectory $i$

 1: Init the GPU device
 2: Memory allocation on the GPU of the required structures
 3: $th = a_b$                                                        ▷ threshold
 4: for $i \leftarrow 1$ to $ntraj$ do
 5:     $\vec{T}[i][0] \leftarrow (0,0,0)$          ▷ Init tracer trajectory at origin of coordinates
 6:     $\vec{Pos}[], \vec{Vel}[] \leftarrow Init\_Part(\vec{Pos}[], \vec{Vel}[])$   ▷ Init locations and velocities of $N$ particles
 7:     Copy the values of $\vec{Pos}[]$, $a[]$ and $NList[][]$ from CPU to GPU
 8:     $NList[][], th \leftarrow$ cuNeighbors($N, \vec{Pos}[], \vec{Vel}[], a[], mn, NList[][], th$)
 9:     for $t \leftarrow 0$ to $ttraj$ do
          Difus algorithm (lines 10-18)
10:         Add Brownian kick in $\vec{Pos}[]$ on the CPU
11:         Copy the values of $\vec{Pos}[]$ and $\vec{Vel}[]$ from CPU to GPU Numerical solution of the Langevin equation
12:         $\vec{df}[] \leftarrow$ cuForces($N, \vec{Pos}[], a[], NList[][]$)
13:         $\vec{Pos}[], \vec{Vel}[] \leftarrow$ cuUpPosVel1($N, \vec{Pos}[], \vec{Vel}[], \vec{df}, \gamma_0, \delta t$)
14:         $\vec{df}[] \leftarrow$ cuForces($N, \vec{Pos}[], \vec{Vel}[], a[]$)
15:         $\vec{Vel}[] \leftarrow$ cuUpPosVel2 ($N, \vec{Vel}[], \vec{df}[], \delta t$) Corrections to fix the center of mass
16:         $\vec{V}_{mc} \leftarrow$ cuVelVMC($N, \vec{Vel}[]$)
17:         $\vec{Vel}[] \leftarrow$ cuUpVelVMC ($N, \vec{Vel}[], \vec{V}_{mc}$)
18:         Copy the values of $\vec{Pos}[]$ and $\vec{Vel}[]$ from GPU to CPU
19:         if the displacement with respect to the location of $NList$ for at least one particle $\geq th/2$ then
20:             $NList[][], th \leftarrow$ cuNeighbors($N, \vec{Pos}[], \vec{Vel}[], a[], mn, NList[][], th$)
21: Memory deallocation on the GPU
22: return $\vec{T}[i][t]$ with $0 \leq i \leq ntraj$, $0 \leq t \leq ttraj$     ▷ Return tracer trajectories
```

and cuUpVelVMC). According to the system size, each kernel uses an adequate amount of threads in order to accelerate the calculation of the involved operations.

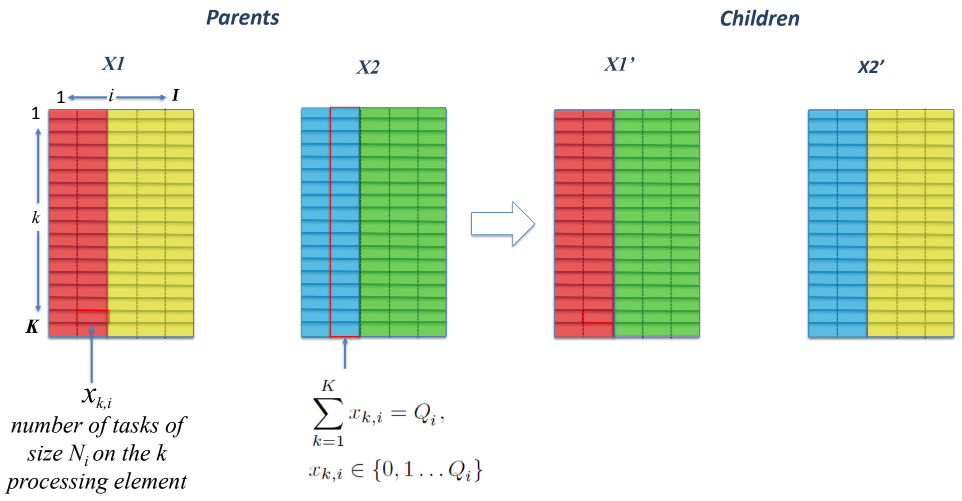
On the other hand, the complexity of the computation related to the neighbors is higher than the computation of the positions and velocities. As a counterpart, the list of neighbors must be updated only if any particle has moved more than a threshold distance. Moreover, a specific definition of neighborhood has been implemented to reduce the computational cost. It is based on two ideas: 1) two particles are neighbors if their distance is smaller than the radius of the particles, and 2) it is enough to consider the 200 closest particles in the list of neighbors of each particle. The list is labelled as $Nlist[i][200]$ (with $0 \leq i < N$) and stores the index of neighbors of each particle. $Nlist$ is stored by column-major order to accelerate its computation on the GPU.

Exploiting 2-3 parallelism levels on heterogeneous clusters

Once we have accelerated the computation of a trajectory (level 1), the next step is to compute sets for several sizes to extrapolate γ_∞ (levels 2 and 3). To compute such sets, several processing units (CPU-cores and GPUs) can be used as the nodes of a cluster. Algorithm 1 for a GPU and also a sequential version for a CPU-core are used here to exploit the different units of the cluster. Let $\sum_{i=1}^{I} Q_i$ be the number of trajectories to compute, where I is the size of the system (we said previously that the possibilities were $N =$ 216, 512, 1000, 2197, 4096, 8000, 15625 and 32768), and $1 \leq i \leq I$. The main idea to accelerate this part is to minimize the makespan, C_{max}:

$$\begin{aligned}
&\text{Find:} && X \quad \text{to} \\
&\text{minimize:} && C_{max} \\
&\text{subject to:} && t_k = \sum_{i=1}^{I} x_{k,i} t_{k,i} \leq C_{max}, 1 \leq k \leq K \\
& && \sum_{k=1}^{K} x_{k,i} = Q_i, 1 \leq i \leq I \\
& && x_{k,i} \in \{0, 1, \dots, Q_i\}, 1 \leq k \leq K; 1 \leq i \leq I
\end{aligned} \tag{1.14}$$

being $x_{k,i}$ (each element of a $k \times i$ matrix X, which is the result) the number of tasks of each size N_i assigned to the processing unit k, and $t_{k,i}$ the runtime to compute a task of size N_i on the processing unit k. Each task must be computed one (and only one) time.



Parents
Children
X1
X2
X1'
X2'
1
i
I
1
k
K
$x_{k,i}$
number of tasks of size N_i on the k processing element
$\sum_{k=1}^{K} x_{k,i} = Q_i,$
$x_{k,i} \in \{0, 1 \dots Q_i\}$

Figure 1.16: Crossover operator is applied over two individuals splitting their matrices in a random column and swapping the right and left parts. As a result, a new pair of individuals is obtained.

A genetic algorithm (GA) [77] was defined to find a scheduling that minimizes C_{max}. The GA takes as population a set of possible solutions, that is, the individuals of populations are X matrices. It starts using a randomly generated population. Each iteration of the GA applies the operators 1) crossover, 2) mutation, 3) evaluation, and 4) selection. In the crossover operation, a random pair of individuals are selected as parents. Two new individuals (children) are obtained from the parents selecting a random column and splitting the matrices of the parents and swapping the right and left parts of such parents as it is shown in Fig. 1.16. After the crossover, a child can suffer a mutation with a probability of 1%. The mutation exchanges tasks of the same size between two processing units. Then, the individuals are evaluated and ordered from best to worst using a fitness function. Finally, only the best individuals are kept.

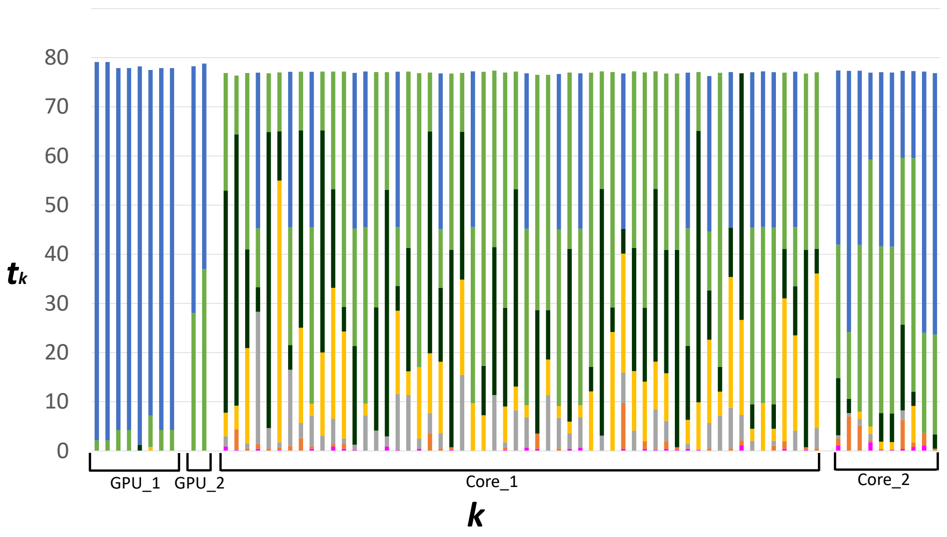


Figure 1.17: Runtime, in hours, for the GA on a cluster composed by 2 kind of CPU's and 2 kind of GPU's (blue - N=15625, green - 8000, black - 4096, yellow - 2197, grey - 1000, orange - 512, pink - 216).

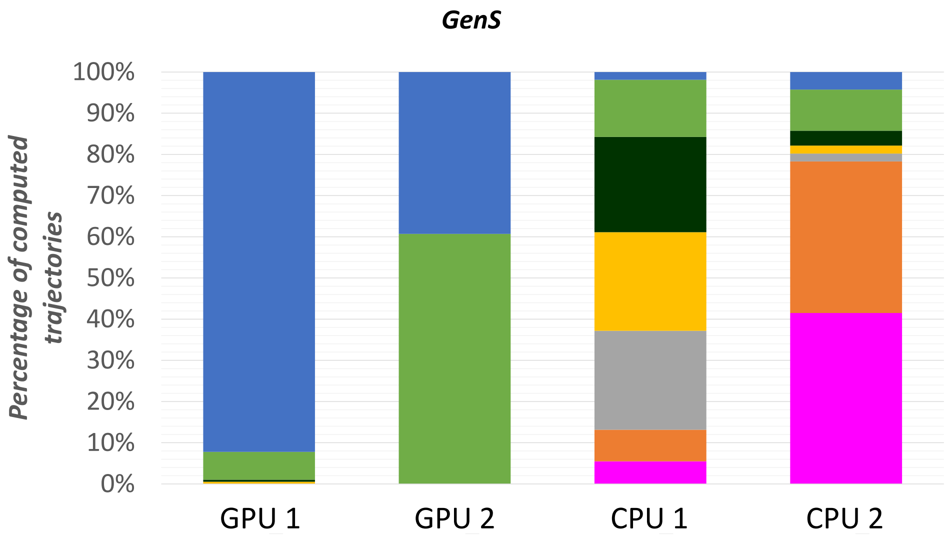


Figure 1.18: Percentage of each type of task solved on each kind of PU (blue - N=15625, green - 8000, black - 4096, yellow - 2197, grey - 1000, orange - 512, pink - 216).

Fig. 1.17 illustrates an example of a solution that the GA gives for a case of 250 trajectories of each one of the sizes using a cluster composed of 56 cores of Bullx R424-E3 Intel Xeon E5

2650 with 8GB RAM, 10 cores of Bullx R421-E4 Intel Xeon E5 2620v2 with 64GB RAM, 8 NVIDIA Tesla M2070 GPUs, and 2 NVIDIA Kepler GK210 GPUs [55]. The figure shows the distribution of tasks in the different machines. It is visible at a glance how the heaviest tasks (labelled in blue) are mainly assigned to the fastest devices. The rest of the tasks are reordered in a way that tries to minimize the makespan while balancing the load. The same result is shown in Fig. 1.18, but such figure represents the percentage of each type of task executed on each processing unit. It is clear that the biggest sizes are computed on the GPUs. In terms of results, this rearrangement translates into a strong improvement in efficiency compared to other alternatives of the state-of-the-art.

Quantum computing

The applicability of quantum computing to accelerate the computation of the neighbors has also been studied. The creation of a neighborhood can be described as a search of a set of particles that satisfy a condition -their distance is less than a threshold value-. Therefore, an adaptation of Grover's algorithm [18] can be applied if an appropriate oracle is designed. In particular, this oracle needs to determine if the (Euclidean) distance between two particles is under the mentioned threshold value. We have designed such an oracle using the most relevant circuits available in the literature, as it is shown in Fig. 1.19. Our oracle (O_ν^μ) marks μ elements from a set of size ν.

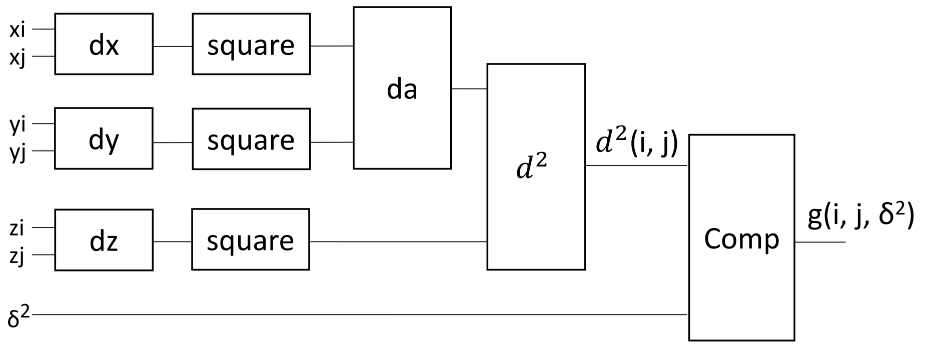


Figure 1.19: Scheme of the final circuit for the $3D$-case. d_x, d_y and d_z are computed using [73](no input carry) ($\overline{\overline{a}+b}$). The squares are computed using [39]. d_a and d' are computed using [73] (no input carry). Finally, the comparison is computed using the circuit of [80].

We have developed three algorithms (three adaptations of Grover's one) that find such particles using only $O\left(\sqrt{\frac{\nu}{\mu}}\right)$ calls to the oracle (a classical algorithm would need $\Omega\left(\frac{\nu}{\mu}\right)$ consults). Similar to the GPU code, these quantum algorithms look for pairs of close particles, and then update the list of neighbors. The first algorithm is focused on the case in which the number of neighbors is previously known. Algorithms 2 and 3 consider that this number is unknown. The difference between the second and the third is that the second one chooses the number of iterations of Grover's algorithm in a uniform way, whereas the third one increases the iterations number, from 1 to $\frac{6}{5}$. These three algorithms can be combined in a meta-algorithm to achieve the computation of the neighbors as follows:

First step: initialize the pairs of close particles

At this initial stage, the parameter ν is initialized as N^2 (all possible pairs of neighbors in the whole system), and μ is the number of close pairs to be found. The choice of the algorithms is as follows:

- **If** μ is not known, then:
  - **If** μ is believed to be negligible in relation to the total number of pairs, use Algorithm 2 ($O(N)$ oracle calls in the worst case) with an estimated upper bound $B \leq 27N$ of μ.
  - **Else**, use Algorithm 3 with an estimated upper bound $B \leq 27N$ of μ ($O(N\sqrt{N})$ oracle calls in the average case).
- **Else** (μ is known), then:
  - **If** μ is negligible in relation to the total number of pairs, use Algorithm 1 (in the worst scenario, $O(N)$ oracle calls) or Algorithm 2 ($O(N \log N)$ oracle calls in the worst case) with $B = \mu$.
  - **Else**, use Algorithm 1 ($O(N\sqrt{N}\log N)$ oracle calls in the worst case) or Algorithm 3 with $B = \mu$ ($O(N\sqrt{N})$ oracle calls in the average case).

Second step: update the set of particles close to fixed ones

At this stage, the parameter ν is initialized as N, the number of updated particles is α, and for a fixed particle, μ represents the number of close particles to be found.

The alternatives are the following:

1. If $\alpha \log \alpha$ is close to N, then backtrack to the first step.
2. Else, set $S = \left\lceil \frac{\log\left(\frac{w}{\alpha}\right)}{\log(w)} \right\rceil$. Then:
   (a) If μ is known, then use Algorithm 1 S times for each of the α particles ($O(\sqrt{N}\sqrt{\alpha}\log \alpha)$ oracle calls in the worst case).
   (b) Else, use Algorithm 2 S times for each of the α particles ($O(\sqrt{N}\sqrt{\alpha}\log \alpha)$ oracle calls in the worst case).

This procedure illustrates how to apply the quantum oracle model of a statistical nature and the advantages of using superposition, but also the difficulty of information retrieval and the construction of circuits that process many qbits. In any case, the results obtained have been better than those expected from the asymptotic analysis, especially in cases where the particle density is low.

This work has been presented in the 21th International Conference Computational and Mathematical Methods in Science and Engineering, and is being revised by an international journal as part of a special issue dedicated to the mentioned conference.

1.4.2 MultiDimensional Scaling

Dimensionality reduction methods are focused on mapping high-dimensional real-world data into lower-dimensional spaces [6]. The objective is to minimize the size of the data but without losing the information it keeps. There is a wide amount of dimensionality reduction methods. Among them, the so-called MultiDimensional Scaling (MDS) methods are very popular [14].

Algorithm 2 SMACOF (m, s, Δ, $kmax$, ε, Y)

```
Require:
    $m$: number of items;
    $s$: dimension of low-dimensional space;
    $\Delta$: $m \times m$ matrix of dissimilarities of observed data on the high-dimensional space ($n$);
    $kmax$: maximum number of iterations;
    $\varepsilon$: threshold for the stress variance
Ensure:
    $Y$: set of finding points in the low-dimensional space stored in a $m \times s$ matrix
 1: Initial Solution randomly generated, $Y^0$
 2: Compute Euclidean distances, $D^0 = [d(Y_i^0, Y_j^0)]$                          ▷ $O(m^2 s)$
 3: $k = 0$, $error = 1$
 4: if $(k < kmax)$ and $(error > \varepsilon)$ then
 5:     Compute Guttman transform matrix, $B^k \equiv B^k(\Delta, D^{k-1})$ (see Alg. 3)    ▷ $O(m^2)$
 6:     Compute Guttman transform, $Y^k = 1/m \cdot B^k \cdot Y^{k-1}$                ▷ $O(m^2 s)$
 7:     Update distances $D^k = [d(Y_i^k, Y_j^k)]$                                   ▷ $O(m^2 s)$
 8:     Compute $E_{MDS}^k$ (Eq. 1.15)
 9:     $error = |E_{MDS}^k - E_{MDS}^{k-1}|$
10:     $k = k + 1$
11: return $Y$
```

MDS techniques try to find a set of points $Y_1, Y_2, \ldots, Y_m \equiv Y$ (m is the number of observations) in the space $\mathbb{R}^s$, $s < n$, whose distances between them are as similar as possible to the distances between the original set of points $X_1, X_2, \ldots, X_m \equiv X$ in the space $\mathbb{R}^n$, with $s < n$ [6]. To obtain such points, these methods minimize the stress function:

$$E_{MDS} = \sum_{i<j} \Big(\delta_{ij} - d(Y_i, Y_j) \Big)^2 \tag{1.15}$$

where δ y d are the distances between original and obtained points, respectively. MDS methods are often used to evaluate criteria of objects classification, for graphical visualization, to discover human patterns in psychology, and in a multitude of applications where the correlation between the features of the data is linear [11].

The algorithm called Scaling by MAjorizing a COmplicated Function (SMACOF) is considered the most accurate MDS method. However, it is also the most expensive among them, with a complexity of $O(m^2)$ [11]. SMACOF uses the concept of majorization to minimize the stress function. Majorization consists of approximating a complex function using a simpler one through an iterative process. Each iteration gets a new function over the original one, touching it at the supporting point. Each new function is closer of the minimum of the stress function. This process is shown in Algorithm 2.

The algorithm shows that SMACOF involves large data structures: an input $m \times m$ matrix Δ, several $m \times s$ matrices to save the output, and three auxiliary $m \times m$ matrices to store the similarities among the objects of the low-dimensional space. The number of floating point operations of SMACOF is: $3s/2m^2 + 3s/2m$ for the initialization and $(7/2s + 3/2)m^2 + 1/2(3s + 1)m$ for the iterative process. The computational cost of SMACOF is $O(sm^2)$. The requirements in terms of memory are $O(m^2)$. These requirements have limited the applicability of SMACOF with big datasets. There are several MDS techniques that simplifies this process, but at the cost of reducing the accuracy.

Algorithm 3 $B^k \equiv B^k(\Delta, D^{k-1})$ (Eq. 8.24 of [6] to compute Guttman transform)

```
Require:
    m: number of items;
    Δ: [δ_ij], m × m matrix of dissimilarities based on observed data;
    D: [d_ij], Euclidean distances matrix
Ensure:
    B: [b_ij], Guttman transform matrix
1: for $i = 0; i < m; i++$ do
2:     for $j = i + 1; j < m; j++$ do
3:         if $d_{ij} \neq 0$ then
4:             $b_{ij} = -\delta_{ij}/d_{ij}$
5:         else
6:             $b_{ij} = 0$
7: for $i = 0; i < m; i++$ do
8:     $b_{ii} = -\sum_{j=1, j\neq i}^{m} b_{ij}$
9: return B
```

Parallel Implementations and energy efficiency of SMACOF

We have developed two parallel implementations of SMACOF -multicore and CPU- (extensive description of our both implementations are in [47]). The two implementations focus on the computation of the Euclidean distances, and in the computation of the Guttman transform (Alg. 3). Matrices B^k, D^k and δ are symmetric matrices, so only $L = (m(m+1)/2)$ elements needs to be computed. In such cases, we work with unidimensional vectors of L elements to get a better distribution among the processing units.

The multicore version has been implemented in C and OpenMP. The Intel MKL library has also been used [75]. The L elements of the matrices have been distributed among the cores, synchronizing certain parts of B^k since in this cases the non-diagonal elements must be computed using the rest of the values. The MKL library is used to compute the matrix-matrix product related to the Guttman transform. The GPU version has been implemented in C and CUDA. Three kernels has been defined, one for the computation of the Euclidean distances, and two for the Guttman transform: one for the non-diagonal elements, and a second one to compute the diagonal ones. cuBLAS library and shuffle instructions have been used to accelerate certain parts of such computations [8, 42].

A heuristic for selecting the optimal version and configuration of SMACOF for any specific situation has also been designed and implemented in Python. The heuristic is focused on the optimization of the energy efficiency (EE), which can be described as the ratio of the computational speed and the power. It can be expressed as:

$$EE(r_k) = \frac{F}{T^k(r_k)P^k(r_k)} = \frac{F}{(\frac{T^k(1)}{r_k} + TC^k(r_k))(P^k_{idle} + r_k p^k(r_k))} \tag{1.16}$$

Here, F is the number of floating point operations on one platform k, $T^k(r_k)$ and $P^k(r_k)$ are the runtime and power consumption on r_k machines respectively, $TC^k(r_k)$ represents the runtime penalties due to the contention among the actives machines on the k platform, P^k_{idle} represents

Algorithm 4 Heuristic for computing the set of optimal platforms $\{k_o\}$, with their configurations $\{r^o_{k_o}\}$, which optimize the EE of SMACOF.

```
Require:
    $\mathscr{F} = \{\mathscr{F}^k\}_{k=1}^f$ with $\mathscr{F}^k = \{M_i^k\}_{i=1}^{c_k}$;                ▷ Set of platforms
    Parallel versions of SMACOF($m$, $s$, $\Delta$, $kmax$, $\varepsilon$, $Y$) to execute on the $f$ available platforms;
    $m$ (items), $s$ (output dimensions);                ▷ Particular data size
    $sampling$.
Ensure:
    $\{k_o, r^o_{k_o}\}$ optimize the EE on the $f$ available platforms
 1: Evaluate the number of FLOAT operations of SMACOF($m$, $s$, $\Delta$, $kmax$, $\varepsilon$, $Y$)
 2: for $k \leftarrow 1$ to $f$ do
 3:     Execute Parallel SMACOF($m$, $s$, $\Delta$, $kmax$, $\varepsilon$, $Y$) on $r_k = c_k$ machines and evaluate its EE denoted by $\mathscr{E}\mathscr{E}^k$
 4:     for $i \leftarrow c_k - sampling$ to $sampling$ do
 5:         Execute Parallel SMACOF($m$, $s$, $\Delta$, $kmax$, $\varepsilon$, $Y$) on $r_k = i$ machines and evaluate $\mathscr{E}\mathscr{E}^{Aux}$
 6:         if $\mathscr{E}\mathscr{E}^{Aux} \leq \mathscr{E}\mathscr{E}^k$ then
 7:             $r_{k_o} = i + sampling$
 8:             Break $i$-loop
 9:         else
10:             $\mathscr{E}\mathscr{E}^k = \mathscr{E}\mathscr{E}^{Aux}$
11: Select the platforms $\{k_o\}$ with their optimal configurations $\{r^o_{k_o}\}$ which maximize EE
12: return $\{k_o, r^o_{k_o}\}$
```

the idle power consumption when no process is actively using any machine and $p^k(r_k)$ is the contribution to the power of every machine.

The heuristic was tested with different sizes of the problem (values are shown in Table 1.6) in three different clusters:

$\mathscr{F}_1$: Bullion S8: 4 Intel Xeon E7 8860v3 (16 × 4 CPU-cores);
$\mathscr{F}_2$: Bullx R421-E4 Intel Xeon E5 2620v2 (12 CPU-cores and 64 GB RAM);
$\mathscr{F}_3$: NVIDIA K80 (composed by two Kepler GK210 GPUs) connected to the host Bullx R421-E4 Intel Xeon E5 2620v2.

Table 1.6 details the test problems used for the evaluation of SMACOF. The runtime, power, and EE of the set of test problems on the first two platforms $\mathscr{F}_1$ and $\mathscr{F}_2$ are shown in Fig. 1.4.2. We also show these values for $\mathscr{F}_3$ in Table 1.7.

Table 1.6: Test problems using several number of items (m), dimensions of multi-dimensional space (n), and dimensions of low-dimensional space (s). The number of iterations of SMACOF used in such tests is 100.

	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$	$T9$	$T10$	$T11$
m	2000	4000	6000	8000	10000	12000	14000	16000	18000	20000	22000
n	100	200	300	400	500	600	700	800	900	1000	1100
s	2	3	4	5	6	7	8	9	10	11	12

As an example, Table 1.8 shows the result of applying Alg. 4 on the two first platforms for the biggest size ($T11$) to automatically select the optimal parallel platform and their best resource configuration. Although the GPU version is faster than the multicore one -as it is described in

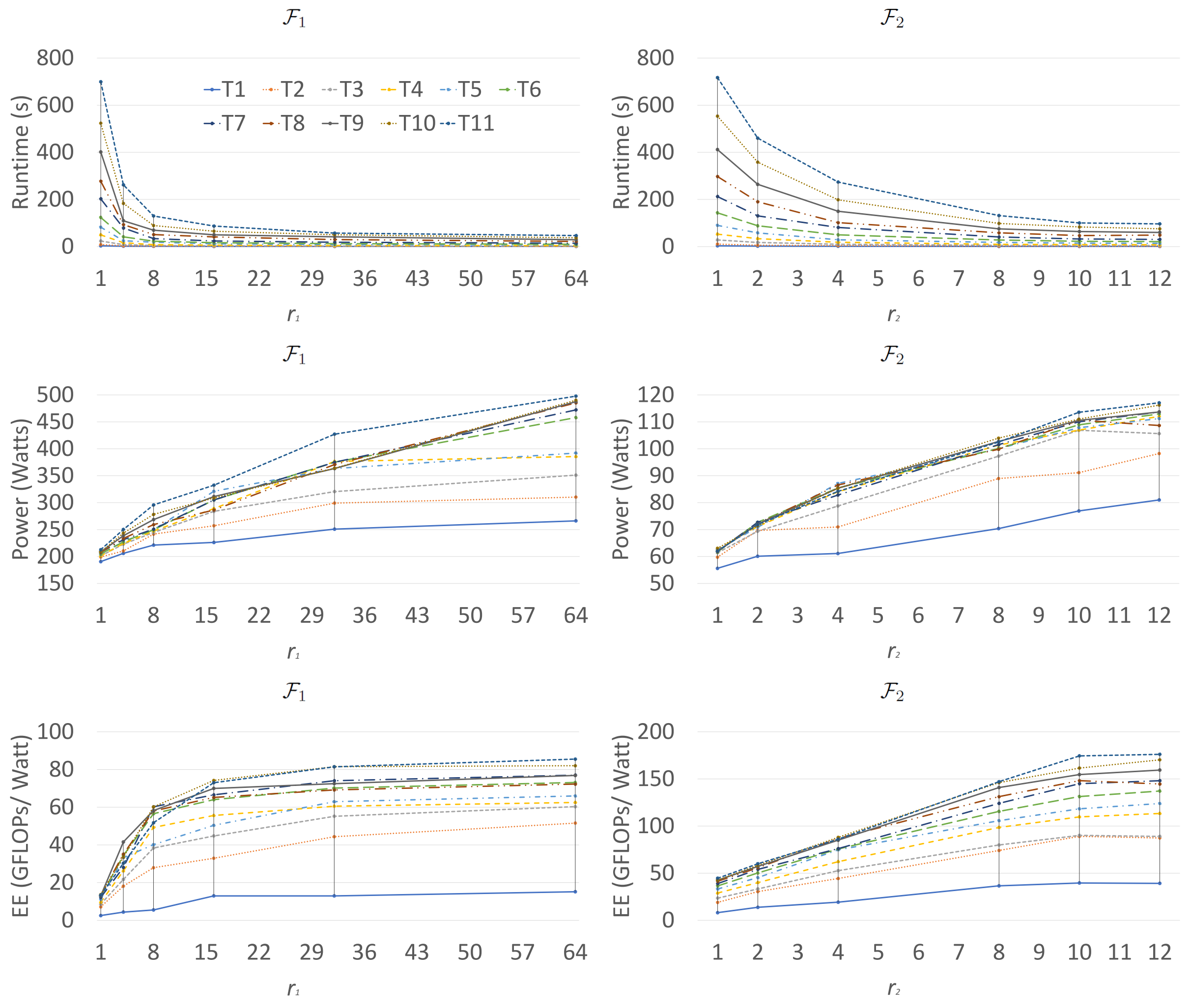


Figure 1.20: Runtime, power, and energy efficiency of the set of test problems on $\mathscr{F}_1$ and $\mathscr{F}_2$ platforms.

our paper [47]-, the multicore version has better EE (each version on the corresponding clusters). Therefore, it can be concluded that in these cases there is no better option, and the selection of the version and cluster depends on the priorities of the users.

1.4.3 Image Processing based on SMACOF and quantum computing

HyperSpectral Images (HSIs) are a special kind of images that contain extended information about the characteristic of the materials across the electromagnetic spectrum [5]. In such images, each pixel can be described as a vector that contains the luminosity of the reflectance value for

Table 1.7: Runtime, power and energy efficiency of the set of test problems (Table 1.6) on $\mathscr{F}_3$ (GPU) platform.

$\mathscr{F}_3$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$	$T9$	$T10$	$T11$
Time (s)	2.8	4.9	5.1	5.9	6.5	11.0	18.8	28.7	42.3	64.9	91.5
Power (Watts)	38.7	98.6	105.2	108.0	112.6	113.6	112.8	110.1	112.3	111.5	110.3
EE (GFLOPs/Watt)	13.8	24.7	75.1	150.0	255.1	257.2	240.7	241.7	228.7	205.9	196.6

Table 1.8: Sampling of EE for $T11$ test according to the benchmarking proposed in Alg. 4 for multicore platforms $\mathscr{F}_1$ and $\mathscr{F}_2$.

$\mathscr{F}_1$		
r_1	64	61
EE (GFLOPs/Watt)	85.5	85.0

$\mathscr{F}_2$		
r_2	12	9
EE (GFLOPs/Watt)	176.1	155.8

each spectral band. These bands do not cover only the visible spectrum but also the infrared. The width of a band depends on the sensor, but is usually between 5 and 10 nm. Since each material has its own reflectance profile, every pixel of the image contains a huge amount of information.

The big amount of information that this kind of images contains is its main advantage. However, it is also challenging due to the high computational requirements [60]. To be able to exploit all this information in an efficient way, pixels are usually labelled into classes according to their spectral values. Moreover, HSIs can be compressed into a low-dimensional images [19]. The previous subsection describes how MDS methods can be used to reduce dimensionality. However, this kind of images contains relations between their data that can not be linearly obtained, so such methods are not valid to maintain the information of HSIs.

There is another well-known dimensionality reduction technique called ISOmetric MAPping (ISOMAP) that generalizes MDS to a non-linear context, replacing Euclidean distances with geodesic ones (Fig. 1.21) [2]. ISOMAP consists of 3 steps:

- A number l of neighbors is set. Then, the algorithm looks for the l nearest points for every point X_i, building a graph G. This can be done, for example, using the K-Nearest Neighbor (KNN) algorithm [82].
- The shortest path between each pair of points in G is computed, that is, $d(X_i,X_j) = min\{d_G(X_i,X_j), d_G(X_i,X_n) + d_G(X_n,X_j)\}$, for each pair of point X_i and X_j. This can be done using Dijkstra's algorithms [24].
- An MDS method is used at this point for dimensionality reduction.

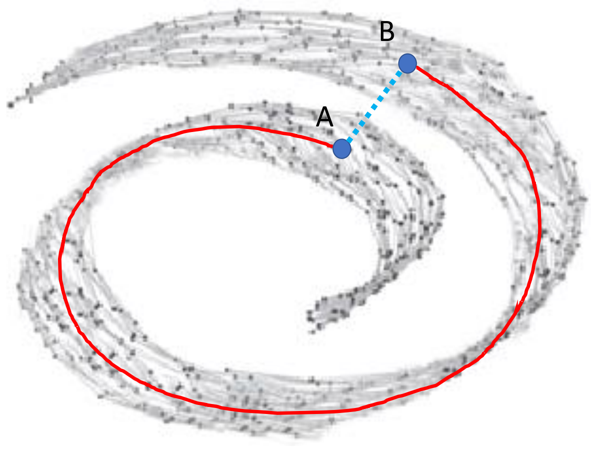


Figure 1.21: Line red represents the geodesic distance between A and B. In this structure, the distance between points depends of its form, being Euclidean distances (blue line) invalid to give valuable information about the real distance.

ISOMAP often employs eigen-decomposition [78] as the MDS method in the third step since this method (eigen-decomposition) has low computational costs. We have experimentally proven that the use of SMACOF improves the accuracy of ISOMAP in the analysis of HSIs as we present in our paper [50]. Table 1.9 shows the improvement (in terms of accuracy) of using ISOMAP with SMACOF instead of classical MDS (labelled as Eigen-decomposition in the table) in three different HSIs and setting the final dimension (s) in a range from 50 to 10. Moreover, obtained results showed that this version of ISOMAP improves the accuracy of other popular methods like Support Vector Machine, KNN, and Random Forest, used as classificators. It is necessary to deal with higher requirements, but the version of SMACOF exposed in the previous subsection (Subsection 1.4.2) has helped with such requirements.

		SMACOF			**EIGEN-DECOMPOSITION**		
IMAGE	k' / s	**1**	**3**	**5**	**1**	**3**	**5**
Indian Pines	**50**	*0.8112*	0.7958	0.7943	*0.7250*	0.6956	0.6881
	40	*0.8046*	0.7987	0.7912	*0.7200*	0.6965	0.6884
	30	*0.8068*	0.7849	0.7814	*0.7150*	0.6933	0.6893
	20	*0.8179*	0.8069	0.7845	*0.7150*	0.6916	0.6879
	10	*0.8090*	0.7915	0.7877	*0.7050*	0.6896	0.6880
Salinas-A	**50**	*0.9946*	0.9931	0.9890	0.9899	0.9714	0.9658
	40	*0.9952*	0.9913	0.9925	*0.9896*	0.9733	0.9654
	30	*0.9950*	0.9935	0.9904	*0.9898*	0.9765	0.9645
	20	*0.9952*	0.9917	0.9914	*0.9892*	0.9743	0.9699
	10	*0.9963*	0.9890	0.9924	*0.9890*	0.9765	0.9687
Pavia	**50**	*0.9917*	0.9503	0.9488	*0.9729*	0.9365	0.9211
	40	*0.9929*	0.9407	0.9463	*0.9720*	0.9320	0.9232
	30	*0.9940*	0.9597	0.9525	*0.9729*	0.9365	0.9235
	20	*0.9937*	0.9598	0.9526	*0.9735*	0.9312	0.9245
	10	*0.9934*	0.9615	0.9576	*0.9715*	0.9348	0.9234

Table 1.9: Classification results of three wide known test HSI images (Indian Pines, Salinas-A and Pavia) using KNN for $k' = 1, 3$ and 5, and varying the final dimensions in a range from 50 to 10. For a detailed description, please consult our paper [49].

On the other hand, we have also applied quantum computing to certain parts of the reduction of the HSIs. For instance, the problem described in the second step of ISOMAP is similar to the one we are dealing with in microrheology. However, the lack of optimized circuits to perform operations not only on hyperspectral images, but also on the most basic images [33], keeps us working only on this topic at a theoretical level. In order to be able to perform the operations we need, we have started to design some quantum circuits that allow us to test our models at the level of current quantum computers and simulators. One of these operations, which in fact is the first one we have been able to complete successfully, is to perform image binarization. Therefore, not only HSI classification is studied in image processing but also image binarization using quantum computing (Fig. 1.22). Binarization consists of transforming a color image into a black and white one. In a little more detail, what it does is to compare each pixel of the image with a threshold value, setting the pixel to black or white according to the result obtained in this comparison.

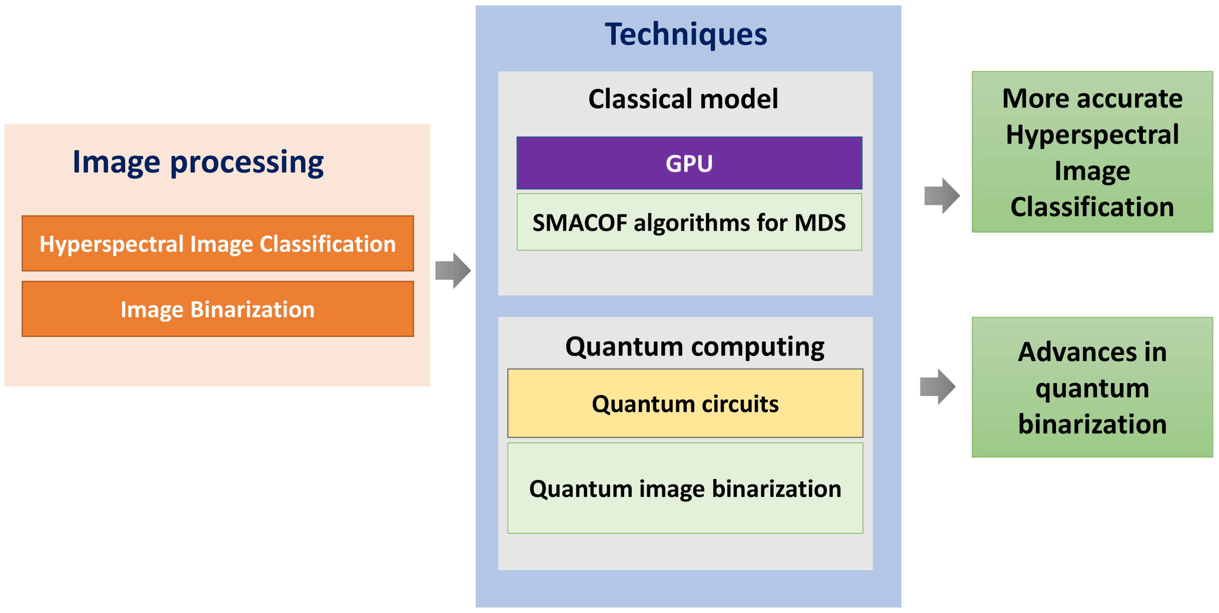


Image processing
Hyperspectral Image Classification
Image Binarization
Techniques
Classical model
GPU
SMACOF algorithms for MDS
Quantum computing
Quantum circuits
Quantum image binarization
More accurate Hyperspectral Image Classification
Advances in quantum binarization

Figure 1.22: HSI classification and image binarization are studied as part of this thesis using both classical and quantum techniques.

The designed circuit employs the NEQR representation (Novel Enhanced Quantum Representation) to work with images. NEQR representation defines a pixel as follows [83]:

$$|C_{YX}\rangle = \frac{1}{2^n}\sum_{Y=0}^{2^n-1}\sum_{X=0}^{2^n-1}\left|C_{YX}^{q-1}C_{YX}^{q-2}...C_{YX}^{1}C_{YX}^{0}\right\rangle \otimes |YX\rangle \tag{1.17}$$

where $\left|C_{YX}^{q-1}C_{YX}^{q-2}...C_{YX}^{1}C_{YX}^{0}\right\rangle$ codifies the value of the pixel (X,Y), n is related to the size of the image, and q defines the color range. The design of the circuit is based on a circuit available in the literature [80]. The original circuit is shown in Fig. 1.23. It consists of two distinct parts: a first part that performs the comparison between each pixel and a threshold value set in advance, and a second circuit that will set the pixel as white or black depending on the result of the comparison. It compares the pixel to that threshold value and returns 0 if it is less, or 1 if it is greater or equal. This original circuit uses a comparator that, although it was the most optimized in cost and noise tolerance, was still not feasible for implementation in today's quantum computers, or even in a simulator without having to resort to the power of a cluster (and even then it is only suitable for the smallest prototypes).

To improve the original circuit, we designed two new comparators and published them in [49]. Both have a common goal: to be fault tolerant but involving the lowest possible cost. However, the second one also tries to maximize speed, being a prototype that although it is not so useful today (in a context where gaining speed at the cost of increasing size and cost is not feasible), we are confident that it will be useful in the future when quantum technology advances. To achieve the goal of reducing the cost, we resort to a novel gate called temporary logical-AND. This gate allows us to perform a proprietary implementation of the most optimized addition algorithms for computer architecture, considerably reducing the aforementioned cost. Starting from the idea that a semi-comparator is enough and a full comparator is not necessary, the comparison is performed as a subtraction $A-B$, knowing that if the result is negative it will imply that A is greater, and that otherwise A is less or equal. This subtraction, similar to a classical computer, is

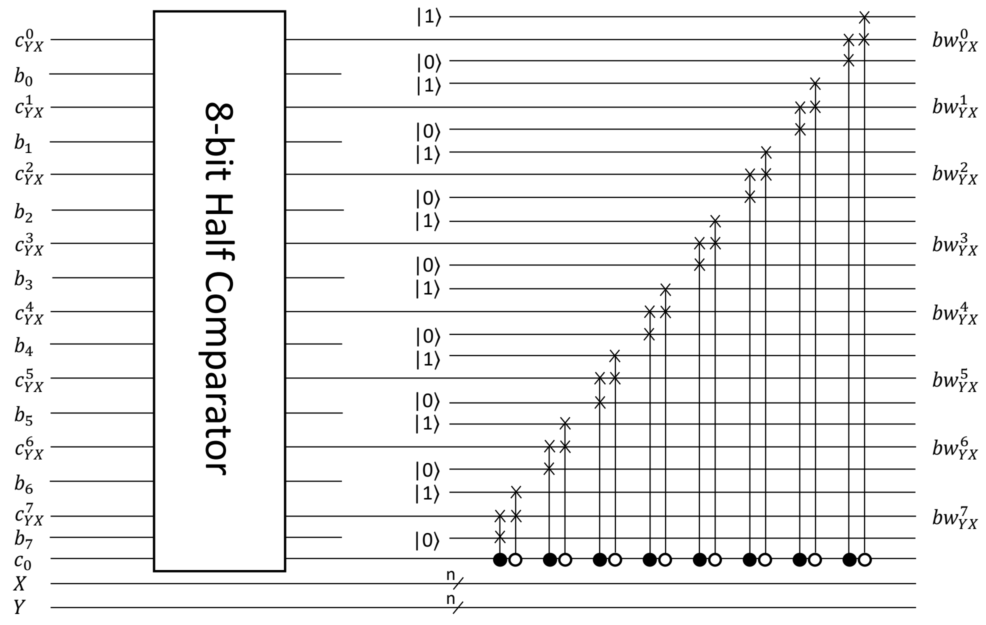


Figure 1.23: Circuit for image binarization proposed in [80].

performed in the form $A-B$. The output of this circuit can be connected directly to the second part of the original circuit, reducing the cost of the original circuit by almost three times. Table 1.10 shows a comparison in terms of T-count, T-depth and ancilla qubit of our proposals versus other state-of-the-art implementations. Our proposed comparators improve the existing ones in terms of T-count and T-depth. Our first comparator greatly improves the T-count compared to existing circuits, while the latter is the only existing log-order comparator in the literature.

Circuit Comparator	T-count	T-depth	Ancilla qubit
Xia et al. (2018) [79]	$14n$	$6n$	2
Xia et al. (2019) [80]	$14n-7$	$6n-3$	2
Li et al. (2020) [29]	$14n-7$	$6n-3$	1
Proposed comparator	$4n$	$2n$	n
Proposed comparator	$12n-8W(n)-4Log(n)$	$Log(n)$	$4n-2W(n)-2log(n)$

Table 1.10: Analysis of comparators in terms of T gates and ancilla qubits for a number of n binary digits. $W(n)$ represents the number of ones in the general expansion.

Thanks to the use of such comparators, binarization requires fewer resources. Fewer quantum gates means not only lower cost, but also lower probability of error. On the other hand, their higher speed also has a positive influence on error reduction, since we know that qubits lose their value over time. Thus, the proposed comparators contribute to binarization (and in general, to any application that requires such an operation) by greatly reducing the cost, improving speed, and offering good fault tolerance.

2. Contributions to scientific journals

According to the regulations of the International Doctoral School of the University of Almería (EIDUAL), at least three publications are required to write a thesis by compendium modality. Of these three publications, two of them must be included in category A of the evaluation scale of research results contained in the University of Almería's Research and Transfer Plan approved in the corresponding year. A third contribution, different from the previous ones and necessarily in a journal, must be included in category B. In the case of the PhD in Computer Science, the responsible committee establishes as category A the journals with JCR included in the first two quartiles, and category B the journals included in the Q3 quartile.

This thesis is made up of 9 publications, of which 8 would fall within the aforementioned category A, and 1 of them in category B. Their classification according to the Journal Citation Reports (JCR) is as follow:

- **Q1 papers**: 5
- **Q2 papers**: 3
- **Q3 papers**: 1

Following the aforementioned regulations, this chapter presents the articles included in the thesis. In addition to the papers already published, there is a paper produced as a result of the advances achieved in this thesis that is currently in process of being reviewed. This paper is important because it is a link between the HPC techniques used so far to solve the microrheology problems studied in the thesis and the quantum computing techniques also developed during the thesis. That is why, although it is not yet published, I wanted to include it in this chapter.

The articles are presented following the structure developed in chapter 1. Thus, subsection 2.1 presents the progress made in the search for new tools that allow the advancement of quantum computing in the fields addressed in the thesis. The first of the articles included in this subsection is a converter of binary numbers in sign-magnitude to two's complement, which will allow the use of this notation and its benefits (such as being able to perform subtractions faster) in quantum computers and simulators. The converter is focused on increasing the speed of the operation, being still at the time of writing the fastest existing converter. A second paper follows this one and proposes up to three new converters, but this time focused on reducing their cost and minimizing noise effects, being again the most suitable for such purposes. Also included is a paper that performs a review of currently available adders for quantum environments. As it has been mentioned in the previous chapter, addition is one of the most important operations in quantum computing, so a review about this kind of circuits it was necessary.

Section 2.2 details the advances made in microrheology using HPC and quantum computing. This includes a paper presenting a simulation of active microrheology in hard colloids. The second paper, which continues the study initiated by the first one, studies the dynamics of a large tracer pulled with a small force in a bath of quasihard colloidal spheres using Langevin dynamics simulations. The third article details the genetic algorithm used to perform the scheduling of the thousands of tasks that had to be computed to arrive at the results of the two previously mentioned articles. Finally, the fourth article -the one that is submitted but not yet published- uses quantum computing to perform (and accelerate) the computation of neighboring particles in the studied systems.

The third section (2.3) shows our work related to the dimensionality reduction and digital image processing. Our article proposes two optimized versions -with CUDA and OpenMP, respectively- of SMACOF, a method of high computational cost that allows to reduce the dimensionality of a large amount of data. Moreover, a heuristic for selecting the optimal version and configuration of SMACOF for any specific situation that optimizes the energy efficiency is also presented.

Finally, the fourth section (2.4) shows our advances in Image Processing. Our first article shows, precisely, the use of SMACOF within a larger method called ISOMAP. ISOMAP is then used to reduce hyperspectral images, improving previous results with respect to other methods used in the literature. ISOMAP is also focused on dimensionality reduction, but unlike SMACOF it allows to detect and maintain nonlinear relationships hidden in a datasets, so that all information is maintained after the reduction of such datasets. The second paper shows our advances in quantum computing and digital image processing with the exposition of a fully functional algorithm for image binarization, a fundamental operation to undertake more important tasks in quantum computing.

The articles, although interrelated, have been independently published in their respective journals. That is why each has its own bibliography, which is also in the format of each journal. Therefore, the bibliography of each publication should be consulted in the article itself, and not in the bibliography section of this thesis, which only includes the bibliography presented in the other sections of the document.

Finally, it should be clarified that as a direct or indirect consequence of this thesis, other works such as contributions to congresses or articles in non-indexed journals have been published. Since in this section only the merits leading to the thesis by compendium are presented, these contributions are shown, in a summarized form, in a later chapter.

2.1 Quantum computing tools

As it has been mentioned, this section contains the following papers:

F. Orts, G. Ortega and E.M. Garzón. An optimized quantum circuit for converting from sign-magnitude to two's complement. *Quantum Information Processing*, 18(332), 1-14, 2019. JCR (2019) = 2.433. Subject categories = Physics, Mathematical: 7/55 (**Q1**); Physics, Multidisciplinary: 34/85 (Q2); Quantum Science & Technology: 9/17 (Q2).

F. Orts, G. Ortega and E.M. Garzón. Efficient reversible quantum design of sign-magnitude to two's complement converters. *Quantum Information and Computation*, 20(9 & 10), 747-765, 2020. JCR (2020) = 0.976. Subject categories = Computer Science, Theory & Methods: 82/110 (**Q3**); Physics, Mathematical: 42/55 (Q4); Quantum Science & Technology: 15/17 (Q4); Physics, Particles & Fields: 26/29 (Q4).

F. Orts, G. Ortega, E.F. Combarro and E.M. Garzón. A review on reversible quantum adders. *Journal of Network and Computer Applications*, 170(102810), 1-16, 2020. JCR (2020) = 6.281. Subject categories = Computer Science, Hardware & Architecture: 5/53 (**Q1**); Computer Science, Interdisciplinary Applications: 14/112 (Q1); Computer Science, Software Engineering: 6/108 (Q1).

2.1.1 An optimized quantum circuit for converting from sign-magnitude to two's complement

Title	An optimized quantum circuit for converting from sign–magnitude to two's complement		
Authors	*F. Orts, G. Ortega, E.M. Garzón*		
Journal	Quantum Information Processing		
Year	2019		
Volume	18(332)		
Pages	1-14		
DOI	https://doi.org/10.1007/s11128-019-2447-7		
IF (JCR 2019)	2.433		
Categories	Physics, Mathematical:	7/55	**(Q1)**
	Physics, Multidisciplinary:	34/85	(Q2)
	Quantum Science & Technology	9/17	(Q3)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

Quantum Information Processing (2019) 18:332
https://doi.org/10.1007/s11128-019-2447-7

An optimized quantum circuit for converting from sign–magnitude to two's complement

F. Orts[1] · G. Ortega[2] · E. M. Garzón[1]

Received: 10 November 2018 / Accepted: 5 September 2019 / Published online: 11 September 2019

Abstract
Nowadays, one of the critical issues to implement quantum algorithms is the required number of elementary gates, qubits and delay. Current quantum computers and simulators are mainly prototypes, and there is a lack of computational resources. Therefore, it is necessary to optimize the quantum operations to reduce the necessary number of gates and qubits. This work presents a novel reversible circuit which is able to convert signed binary numbers to two's complement of N digits in a quantum environment. The depth of the circuit is O(log N). It is based on the fastest out-of-place carry look-ahead addition quantum circuit currently available. This addition circuit has been adapted to make the conversion using the minimum number of gates and qubits, being faster than other adder circuits. A robust metric has been used to measure the quantum cost, delay, ancilla inputs and garbage outputs of the proposed converter. Moreover, it has been compared with others described in the literature.

Keywords Quantum computation · Quantum circuit · Reversible circuit · Two's complement · Sign–magnitude representation to two's complement converter

1 Introduction

Quantum computers are based on reversible gates as they must satisfy the principles of quantum mechanics, for example, the reversibility [26]. There is a wide variety of the literature about building circuits in quantum computers using reversible gates, especially circuits related to arithmetic operations. For instance, there is a special interest in getting faster arithmetic reversible gates to be used as a module in Shor's algorithm. There are optimized gates to compute addition [6,36], subtraction [25,28, 35], multiplication [4,8] and division [11,27].

F. Orts
francisco.orts@ual.es

1 Informatics Department, University of Almería, ceiA3, Almería, Spain

2 Computer Architecture Department, Campus Teatinos, University of Málaga, Málaga, Spain

However, the optimization of arithmetic gates is not the only way to improve these circuits. Sometimes, it can be done with new high-level approaches, like using different formats to represent the information. For example, the two's complement is the way how classic computers represent integers to simplify the hardware for additions and subtractions [1]. In terms of quantum computers, adder circuits are faster than subtractor ones [35,36]. Focusing our attention on the two's complement, subtractions can be computed as additions.

In this work, a circuit to convert from signed binary numbers to two's complement is presented. The design of a two's complement quantum converter can be based on quantum gates or quantum adders that compute $\overline{a}+1$. Our proposal is based on the most optimized state-of-the-art adder circuits for quantum computers since they improve the converter circuits based on quantum gates in terms of delay. The conversion from a signed binary number, a, to two's complement can be computed as $\overline{a} + 1$ [13]. It can be done negating each digit of a and using an adder to compute $\overline{a} + 1$. The best quantum adders in terms of cost and depth are proposed in [6,36]. They are considered as the start point to design a specific adder to compute $\overline{a} + 1$.

The rest of the work is presented as follows: Sect. 2 details popular metrics to evaluate a quantum circuit. Section 3 describes the state-of-the-art converter circuits. Section 4 presents the proposed circuit, and Sect. 5 compares the proposed circuit with respect to the state-of-the-art converter circuits described in Sect. 3. Finally, Sect. 6 summarizes the conclusions.

2 Measures in a quantum circuit

The metric described in [22] has been adopted in this work. This metric defines four important factors to measure a circuit in terms of efficiency:

- Number of ancilla inputs: constant inputs used to perform auxiliary operations.
- Garbage outputs: outputs which cannot be used at the end of the circuit since it is impossible to know their values. Unless these garbage outputs were reversibly removed (uncomputed), such outputs (qubits) may not be used later, which would result in a waste of resources. So, if they were entangled with inputs of other circuits, they would produce uncertain results [26].
- Delay: the logical depth of the circuit. It is an important parameter which is related to the efficiency of the circuit [35]. In [22], $\triangle$ is defined as the delay unit.
- Quantum cost: number of gates.

It is necessary to underline that not all gates have a similar size. For instance, it is unfair to consider the Pauli-X gate [41] and the Toffoli gate [38] are similar in terms of quantum cost or delay, as the Toffoli gate involves 5 2×2 gates (two controlled-V gates, one controlled-V^+ gate [9] and two controlled-NOT (CNOT) gates) [26] and Pauli-X gate is one 1×1 gate. So, [22] sets the delay according to the size of the gates. This metric has also been considered in this work. Authors in [22] set the delay of 1×1 and 2×2 gates to $1\triangle$, and the delay of an $N \times N$ gate is calculated as its depth when it is built using 1×1 and 2×2 gates. Moreover, the quantum cost of a circuit depends on its number of gates with delay $1\triangle$. For instance, as Toffoli gate has

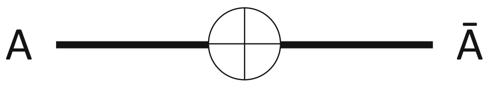


Fig. 1 Symbol of the Pauli-X gate

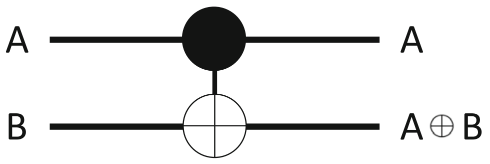


Fig. 2 Symbol of the CNOT gate

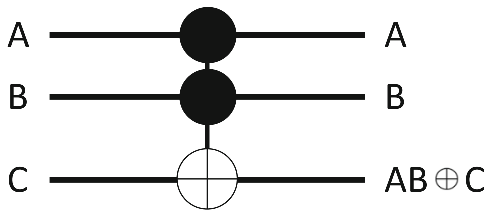


Fig. 3 Symbol of the Toffoli gate

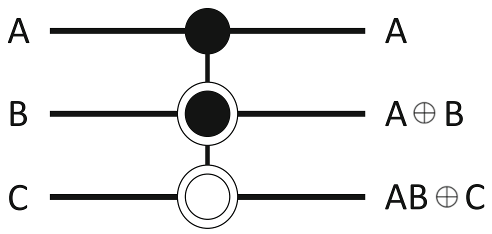


Fig. 4 Symbol of the Peres gate

5 2×2 gates, it has a quantum cost of 5 and a delay of $5\triangle$ (as no operations can be done simultaneously).

It is relevant to underline that different physical realizations have been explored to develop quantum circuits. At those levels of abstraction, the evaluation metrics can also focus on other parameters. For instance, in linear optics, there is a special interest in optimizing the number of controlled-unitary gates since the CNOT gate can only be probabilistically implemented. There are several works focused on optimizing the number of needed CNOT gates to implement the Toffoli gate as it is one of the most used. In [14], authors introduced a new implementation of the Toffoli gate using only two CNOT gates and one generalized controlled-phase gate. Another version of the Toffoli gate was presented in [15], with an optimized controlled-phase gate. This last version has only three two-qubit gates. It is the best option in terms of the number of two-qubit gates.

For the sake of clarity, the symbols of the used gates are shown in Figs. 1, 2, 3 and 4. According to [22], the Pauli-X and the CNOT gates have a quantum cost of 1 and a delay of $1\triangle$. As it was mentioned in the previous paragraph, according to [22], the Toffoli gate has a quantum cost of 5 (three two-qubit gates according to [15]) and a delay of $5\triangle$. The Peres gate is built with two controlled-V^+ gates, a controlled-V gate and a CNOT gate. Therefore, it has a quantum cost of 4 and a delay of $4\triangle$.

3 Methodology to design two's complement converters

The two's complement of an N-digit number is its complement with respect to 2^N. The range of numbers in a two's complement system is $-(2^{N-1}) \leq a \leq (2^{N-1} - 1)$

Table 1 Signed and Two's complement representation of binary numbers with $N = 4$

Signed binary number	Two's complement
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

[13]. For instance, Table 1 shows the conversion of a number a from signed binary to two's complement when $N = 4$. Converting a number a from signed binary to two's complement is as follows. If $a >= 0$, no conversion is necessary because both representations, signed binary and two's complement, of a are equal. However, if $a < 0$, the conversion is necessary. It can be calculated as the inversion of all the digits of a and then to compute $\overline{a} + 1$.

There are two approaches to convert signed binary numbers to two's complement of N digits in a quantum environment. One of them consists of designing a specific circuit for such purpose, and the another one is considering available addition circuits for the conversion.

There are several proposals in the literature which follow the approach based on designing a specific circuit. In [30], authors mathematically propose a new gate, called $SSMT$ gate, to compute the conversion of a 4-digit number. In [2], a quantum gate called TCG is proposed. In a similar way than the previous $SSMT$ gate, it performs the conversion of a 4-digit number ($N = 4$). The TCG gate is also used (and optimized) in [3], and it achieves the best quantum cost (25) for the case $N = 4$. However, it is not possible to join several TCG gates to compute the two's complement of any number with more than four digits. The reason is that the TCG gate does not handle either input or output carries. Besides, it only computes the truth table for the 4-digit case.

Other approaches in the literature are based on using existing addition circuits. This strategy is more widely used because this kind of circuits is more efficient in terms of delay than the specific converter circuits. As it has been mentioned, the conversion can be computed as the inversion of all the digits of a and then $\overline{a} + 1$, so it is only necessary to invert each digit of a with N Pauli-X gates and then to use an existing addition circuit to compute $\overline{a} + 1$. There are several papers about quantum addition circuits of two integers [5,6,19,31–33], which is one of the most important basic operations. The two most efficient addition circuits are [6,36], with a delay of O(log N).

The circuit proposed in this work is based on the circuit presented in [36]. However, instead of using this circuit in a direct way to compute $\overline{a} + 1$, it has been adapted and improved to compute the conversion from signed numbers to two's complement. The details of the proposed circuit are presented in the next section.

4 Proposed two's complement converter

As it has been mentioned in the previous section, the proposal circuit is an adaptation of a reversible out-of-place carry look-ahead adder presented in [36]. The objective is to perform $\overline{a} + 1$ (assuming that a is a negative number in signed binary format). The mentioned adder performs the operation between two numbers, so if we consider a and $b = 1$, it is only necessary to apply a Pauli-X gate to each digit of a at the beginning and the conversion could be done. However, taking into account that b is always 1, several improvements can be done in order to reduce the original circuit.

The converter improves the delay performing g_i and p_i in parallel when possible (being $g_i = a_i b_i$ and $p_i = a_i + b_i$, according to the notation given by [17]) and removing or simplifying the operations related to b, since $b = 1$. It uses N ancilla qubits (Zg_i) to allocate the sum S_{i+1}. It also needs extra ancilla qubits (Zp_i) to store propagated carry values, which are restored to 0 at the end of the circuit to avoid garbage outputs. Since $b = 1$, the qubits used to represent b in [36] are deleted except the least significant one.

The first change to apply with respect to [36] is to include several Pauli-X gates to transform a into $\overline{a}$. The second change consists of removing the first Toffoli gate (quantum cost 5 according to [26], three two-qubit gates according to [15]) and replacing it with a CNOT gate (quantum cost 1). This can be done as the Toffoli gate computes $a_0 b_0 \oplus Zg_0$, which is always $a_0 \oplus Zg_0$ if $b_0 = 1$. Thirdly, the $(N - 1)$ Peres gates (quantum cost 5) can be removed from the circuit. The i-Peres gate computes $a_i \oplus b_i$ in its second qubit, which is always a_i if $b_i = 0$. Also, the i-Peres gate computes $a_i b_i \oplus Zg_i$ in its third qubit, which is always Zg_i if $b_i = 0$. The Peres gates have a quantum cost of 5, so this step reduces the quantum cost in 5 for each removed gate, that is, $5(N - 1)$. Moreover, [36] applies Toffoli gates at locations Zg_i, b_i and Zg_{i+1} for $i = 2$ to $N/2$. All these Toffoli gates except the first one can be removed since, for the case $b = 1$, $Zg_i = 0$ when $i > 2$. This reduces the quantum cost in $(N/2 - 1) \times 5$ and the delay in $5\triangle$. (The remaining gates can be computed in parallel.)

As above mentioned, the inputs of b can be removed except the first one, since the other values are always 0. The b_0 digit can be converted into an ancilla input of value 0 inverted with a Pauli gate at the same time that the digits of a. By this way, the number of qubits is reduced by $N - 1$. All the CNOT gates applied at the end to restore the qubits of b can be deleted (except the one which acts over the new ancilla qubit), so a reduction in $N - 1$ CNOT gates is applied. However, as the a_i-inputs have been inverted, a new inversion is required at the end of the circuit to evade garbage outputs. This adds $1\triangle$ to the global delay, but this operation would be also necessary for the original circuit if it worked with two's complement. The remaining gates on which b_i acts as a control qubit are modified so that the new control qubit was a_i.

This way, the idea is to design a reversible look-ahead adder to compute the particular addition $\overline{a}+1$. The recurrence relation of a general look-ahead adder to accelerate the computation of the carries is well-known [17]. It can be simplified for our particular adder, and the addition can be computed as follows:

$$\begin{aligned}
S_0 &= \overline{a_0} \oplus 1 \\
S_1 &= \overline{a_1} \oplus \overline{a_0} \\
&\dots \\
S_n &= \overline{a_N} \oplus \overline{a_0}\,\overline{a_1}\dots\overline{a_{N-1}}
\end{aligned} \tag{1}$$

where the term $C_i = \overline{a_0}\,\overline{a_1}\dots\overline{a_{i-1}}$ represents the carry to compute $S_i = \overline{a_i} \oplus C_i$.

The proposed circuit to compute $\overline{a}+1$ for an example of $N = 8$ is shown in Fig. 5. The design of such circuit can be explained with the following eight steps.

- **Step 1:** The first step is to transform a into $\overline{a}$ and also to specify b. The input b is supplied by an auxiliary qubit ($b_0 = 1$), instead of the N qubits of [36]. These operations can be done using Pauli-X gates and deleting the $N-1$ qubits of b. It consists of $nine$ Pauli-X gates (quantum cost $1 \times 9 = 9$), and it has a delay of $1\triangle$.
- **Step 2:** It is the start point since its output is $\alpha_0 = \overline{a_0}$ which is involved in the generation of all the carry look-ahead formation process. It consists of one CNOT gate (quantum cost 1), and it has a delay of $1\triangle$.
- **Step 3:** The outputs of this step are $\alpha_1 = \overline{a_0}\,\overline{a_1}$, $\alpha_2 = \overline{a_2}\,\overline{a_3}$, $\alpha_3 = \overline{a_4}\,\overline{a_5}$ and $\alpha_4 = \overline{a_6}\,\overline{a_7}$. This way, the carry C_1 is computed and also the intermediate AND operations to compute the following carries. This stage consists of $four$ Toffoli gates (quantum cost $5 \times 4 = 20$ according to [26], two-qubit gates $3 \times 4 = 12$ gates according to [15]), and it has a delay of $5\triangle$. (All the gates of this step can be computed in parallel.)
- **Step 4:** This stage only computes two outputs, $\beta_1 = \overline{a_0}\,\overline{a_1}\,\overline{a_2}\,\overline{a_3}$ and $\beta_2 = \overline{a_4}\,\overline{a_5}\,\overline{a_6}\,\overline{a_7}$. Therefore, $C_4 = \beta_1$ is ready to compute S_4. It consists of two Toffoli gates (quantum cost $5 \times 2 = 10$ according to [26], two-qubit gates $3 \times 2 = 6$ according to [15]) with a delay of $5\triangle$. (The two gates can be computed in parallel.)
- **Step 5:** This step computes $\delta_1 = \overline{a_0}\,\overline{a_1}\dots\overline{a_7}$ and $\delta_2 = \overline{a_0}\,\overline{a_1}\dots\overline{a_5}$. Thus, C_6 and C_{out} are computed. This step involves two Toffoli gates (quantum cost $5 \times 2 = 10$ according to [26], two-qubit gates $3 \times 2 = 6$ according to [15]), and it has a delay of $10\triangle$.
- **Step 6:** In this step, $\theta 1 = \overline{a_0}\,\overline{a_1}\,\overline{a_2}$, $\theta 2 = \overline{a_0}\,\overline{a_1}\dots\overline{a_4}$ and $\theta 3 = \overline{a_0}\,\overline{a_1}\dots\overline{a_6}$ are computed. Hence, $C_3 = \theta 1$, $C_5 = \theta 2$ and $C_7 = \theta 3$ are ready to compute the corresponding sums. Moreover, $R_0 = 0$ is the result of uncomputing β_2, avoiding a garbage output. This step consists of $four$ Toffoli gates (quantum cost $5 \times 4 = 20$ according to [26], two-qubit gates $3 \times 4 = 12$ according to [15]) and a delay of $5\triangle$. (All the gates of this step can be computed in parallel.)
- **Step 7:** $R_1 = 0$, $R_2 = 0$ and $R_3 = 0$ are the results of uncomputing α_2, α_3 and α_4, respectively. Consequently, all the auxiliary inputs have been uncomputed. It consists of $three$ Toffoli gates (quantum cost $5 \times 3 = 15$ according to [26], two-

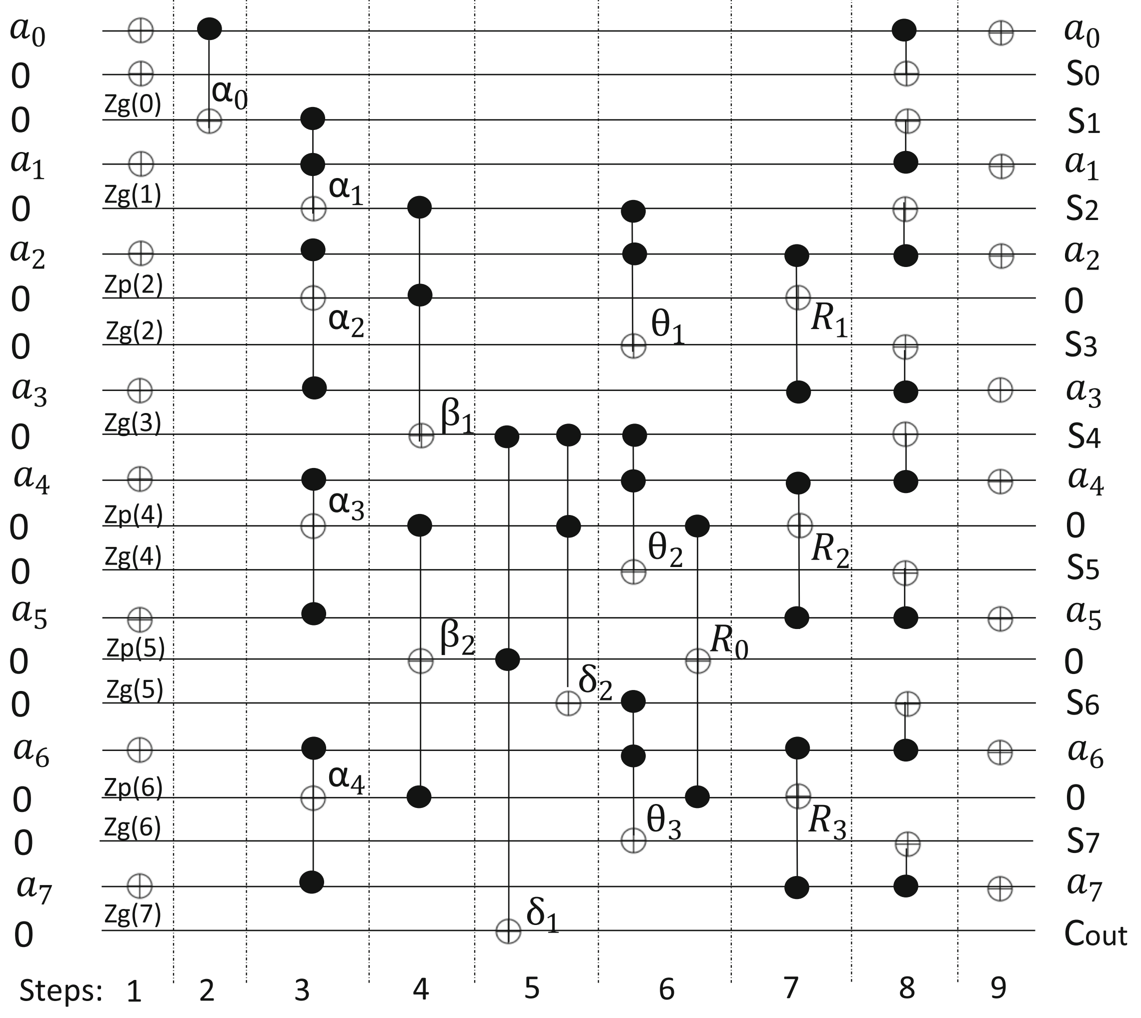


Fig. 5 Design of the converter for a 8-digit number. a_i are the qubits of the input number; $Zg(i)$ will contain S_{i+1} (the result) at the end, and $Zp(i)$ are auxiliary qubits used to store propagated carry value for intermediate digits (they are uncomputed at the end)

qubit gates $3 \times 3 = 9$ according to [15]) and a delay of $5\triangle$. (The *three* Toffoli gates can be computed in parallel.)

- **Step 8:** In this stage, the sums can be calculated since the carries C_i have been computed. So, the output sum is complete at this step. This step consists of *eight* CNOT gates (quantum cost $1 \times 8 = 8$) and a delay of $1\triangle$. (All the CNOT gates can be computed in parallel.)
- **Step 9:** The qubits of a are restored to their input values to avoid garbage outputs. It consists of *eight* Pauli-X gates (quantum cost $1 \times 8 = 8$) and a delay of $1\triangle$. (All the Pauli-X gates can be computed in parallel.)

Therefore, the quantum cost of the converter for $N = 8$ is 101 (54 controlled-unitary gates if the Toffoli gate of [15] is used) and the delay is $34\triangle$. An analogous design can be carried out for a generic N value with a quantum cost of $21N - 15w(N) - 15log(N - 4)$ ($w(N)$ represents the number of ones in the binary expansion of N) and a delay of $logN + logN/3 + 1$.

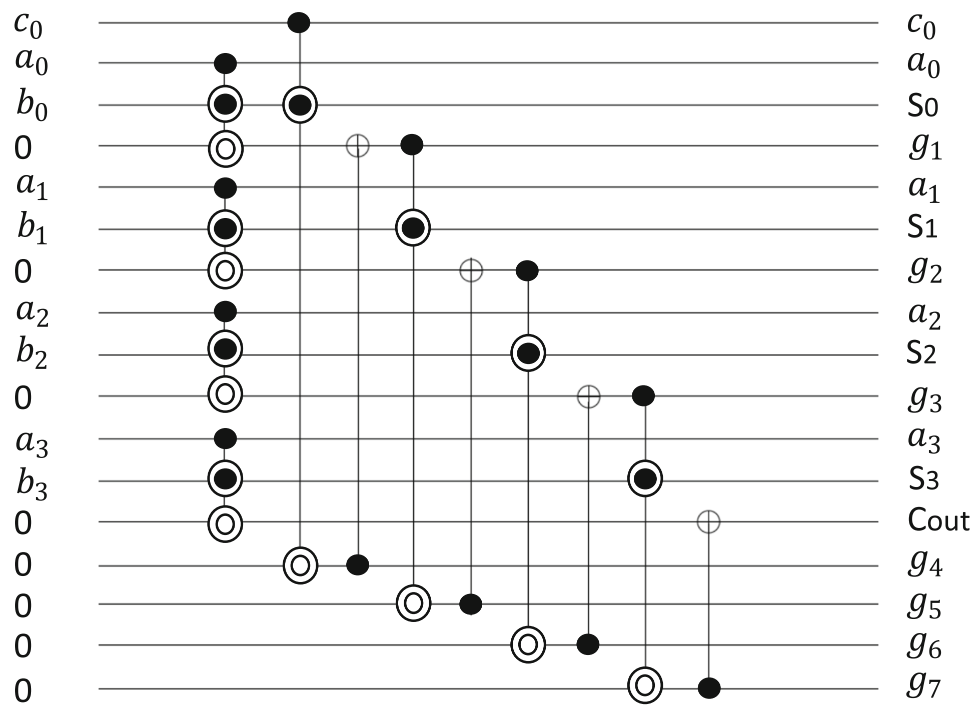


Fig. 6 First carry look-ahead quantum adder design proposed in [29]. a_i and b_i are the input numbers, c_0 is the carry input, S_i are the qubits of the result, C_{out} is the carry output, and g_i are the garbage outputs with an unknown output

5 Results and discussion

In this section, the state-of-the-art circuits are analyzed in order to justify the selection of [36] as the start point to design the converter. This analysis is based on the widely used methodology introduced in [22]. Finally, the obtained results by the proposed converter are compared with the results of the most efficient state-of-the-art circuits.

5.1 Revision of modern quantum adders

After papers [6,36] were published, other adders have been proposed, but none of them has improved them in terms of delay and cost. A reversible 16 digit carry look-ahead adder is proposed in [40]. It consists of 72 CNOT gates and 96 Toffoli gates with a total quantum cost of 552. Several works propose new designs of ripple-carry adder circuits [20,23,39], but the delay of this kind of adders is longer than the carry look-ahead adder circuits [10]. A quantum adder based on genetic algorithms is proposed in [16], which is not comparable since it uses an alternative methodology. The proposal introduced in [34] does not achieve improvements in the terms studied in Sect. 2, but it gives valuable information about fault tolerant techniques in adder circuits. On the other hand, there are several works in which additions and subtractions are computed using the same circuit [7,12,21,24,37]. Although they are able to compute both operations, they are less competitive than [6,36] in terms of delay and quantum cost, due to the extra cost of performing both operations, as it is shown in Table 7 of [24].

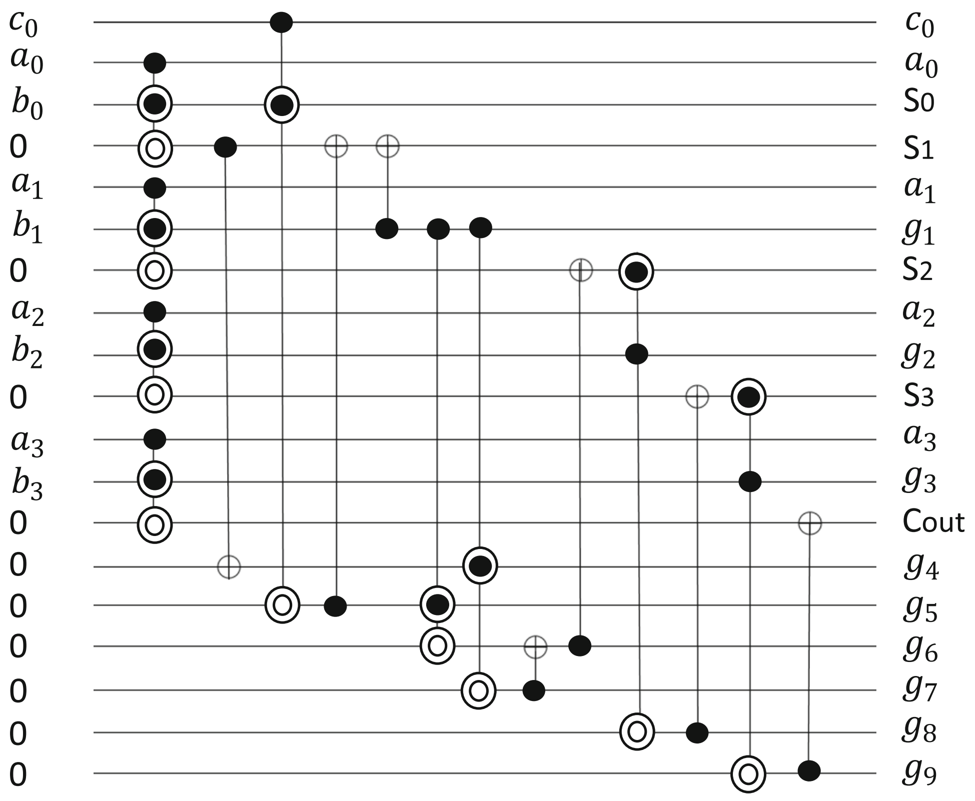


Fig. 7 Second carry look-ahead quantum adder design proposed in [29]. a_i and b_i are the input numbers, c_0 is the carry input, S_i are the qubits of the result, C_{out} is the carry output, and g_i the garbage outputs with an unknown output

In [29], a new adder circuit which optimizes the number of gates and depth is presented (see Fig. 6). It is built using Peres and CNOT gates, with a quantum cost of $2 \times N \times 4 + (N \times 1)$, which is better than the circuit we present even including the extra cost of $2 \times N$ necessary to the inversion and restoration of the a inputs. However, in the proposal of [29], the garbage outputs have not been uncomputed (g outputs in Fig.6). As it was mentioned in Sect. 2, these garbage outputs cannot be used later and it is a waste of resources. The circuit has $3 \times N$ garbage outputs. Notice that uncomputation will increase the number of gates and the depth of the circuit [26], so the numbers given in Table 3 of [29] need to be revised. On the other hand, they consider any output which is not part of the result is garbage output, uncomputed or not. The metrics of [22] does not consider an uncomputed output as a garbage output. According to the metrics proposed in [22], the proposal of [36] overcomes the adder of [29]. They present another circuit which reduces the delay (Fig. 7), but it has the same problems as the previous version.

One more reversible carry look-ahead adder is presented in [18]. It proposes a new technique for generating carry output (see Fig. 8). This circuit uses a new gate called Reversible Partial Adder (RPA) which has a quantum cost of 5 (and delay $5\triangle$) (Figs. 3(a) and 3(b) of [18]) and also has several Fredkin gates, which a cost of 5 each one (and delay $5\triangle$) for the 3 inputs case (Figs. 2(a) and 2(b) of [18]). Figure 8 shows that the quantum cost of the circuit, for the case $N = 4$, includes $four$ CNOT gates

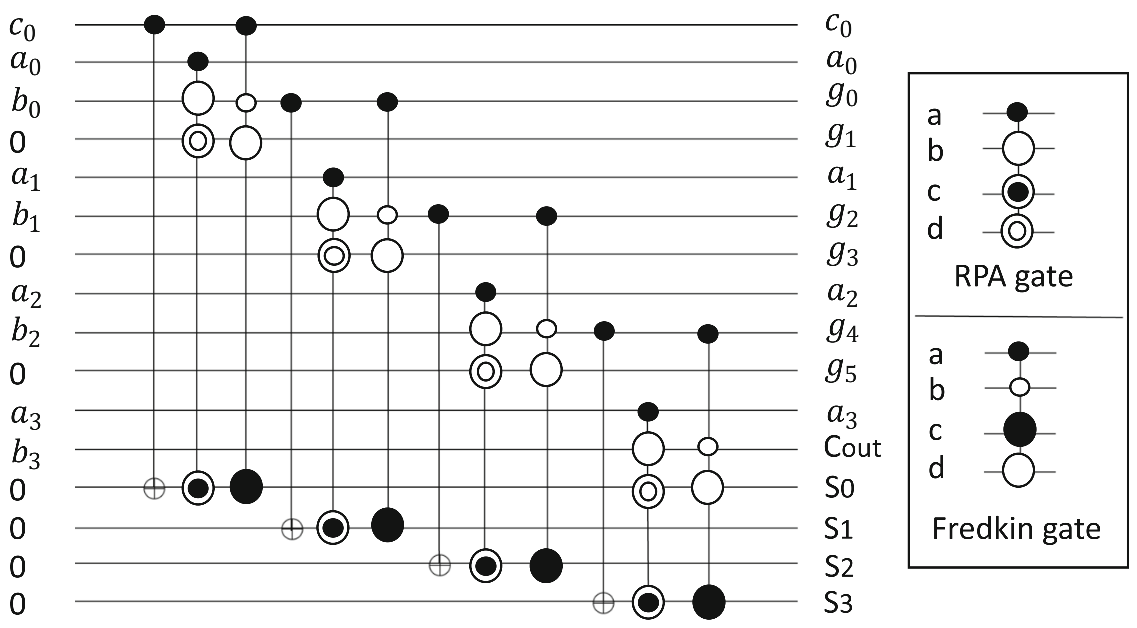


Fig. 8 Reversible carry look-ahead adder proposed in [18]. a_i and b_i are the input numbers, c_0 is the carry input, S_i are the qubits of the result, C_{out} is the carry output, and g_i the garbage outputs

(4 × 1), *four RPA* gates (4 × 5) and *four* Fredkin gates ($4 \times X$), where X is the quantum cost of the 4-input Fredkin gates. (Their quantum cost is not covered by [18].) Considering a quantum cost of 5 for the 4-input Fredkin gate (this is the quantum cost they describe for the 3-input Fredkin gate), the circuit has a quantum cost of $4 \times 1 + 4 \times 5 + 4 \times 5 = 44$. Adding the extra cost of $2 \times N$ to act as a converter, it has a quantum cost of 52. Moreover, the garbage outputs have not been uncomputed like in [29].

5.2 Comparison between the proposed circuit and the most efficient circuits

In this subsection, the results of a comparative analysis between the most efficient circuits to convert signed binary numbers to two's complement of N digits and our proposal are discussed.

Table 2 shows the comparison in terms of quantum costs for the conversion of numbers of N digits for [6], [36] and the proposed circuit. This table takes into account the necessary operations so that the adders of [6] and [36] can make the conversion into two's complement.

The circuit of [6] (year 2004) is less competitive than [36] (year 2013). However, it is still better than the circuits studied in the previous subsection in terms of delay and quantum cost. It has a quantum cost of $28N - 15w(N) - 15log(N) - 6$. The circuit proposed in [36] cannot have input carry, and it has several ancilla inputs to improve the delay and quantum cost. It has no garbage outputs. The complete circuit has $4N - 3w(N) - 3logN$ Toffoli gates (the Toffoli gate has a quantum cost of 5 [26], or *three* two-qubit gates according to [15]), $N - 1$ Peres gates (the Peres gate has a quantum cost of 4 [36]) and $2N$ CNOT gates, that is, it has a quantum cost of $26N - 15w(N) - 15log(N - 4)$. It contains less gates than the circuits presented

Table 2 Comparison of quantum costs for the conversion of numbers of N digits. Works [6,36] include extra cost of $2 \times N$ (N to transform a into $\overline{a}$ and N to restore a and avoid garbage outputs) to act as a converter. The improvement column shows the percentage of improvement in our proposal with respect to [36]

N	[6]	[36]	Proposed circuit	Improvement(%)
4	69	63	41	35
6	105	95	55	42
8	175	160	101	37
10	214	196	119	40
16	399	369	236	36
32	864	802	521	35
64	1869	1743	1166	33
128	3714	3460	2291	34
256	7539	7029	4676	33
512	15,204	14,182	9461	33

Table 3 Comparison of logic depth for N-digit numbers. Works [6,36] include two extra levels (one to transform a into $\overline{a}$ and other to restore a and avoid garbage outputs) to act as a converter

Circuit	Delay $\triangle$	Normal Inputs	Ancilla Inputs	Garbage Outputs
[6]	$logN + logN/3 + 9$	$2N$	$5N/4$	0
[36]	$logN + logN/3 + 4$	$2N$	$5N/4$	0
Proposed circuit	$logN + logN/3 + 1$	N	$5N/4 - 9$	0

in [6]. On the other hand, our proposed circuit improves the quantum cost of [36] thanks to the changes explained in Sect. 4. Results in Table 2 show that the proposed converter improves in terms of quantum costs with respect to the use of general adders to compute the two's complement.

Table 3 shows a comparison of the delay, number of inputs and garbage outputs of some of the most relevant adder circuits in the literature. [36] has a delay of $logN + logN/3+2\triangle$, whereas the circuit of [6] has a delay of $logN+logN/3+7\triangle$. However, to perform $\overline{a}+1$, the inversion of a with Pauli-X gates and another inversion to evade garbage outputs are necessary. This requires two extra levels of depth, so the final delays are $logN + logN/3 + 4\triangle$ and $logN + logN/3 + 9\triangle$, respectively, for [6,36]. In Table 3, the logic depth of the circuits includes the necessary changes to allow them to make the conversion into two's complement. These changes involve to negate a at the beginning and to restore it at the end, to avoid garbage outputs. Furthermore, Table 3 shows that the proposed converter optimizes the number of inputs of the circuit and the answer delay. Therefore, our proposal improves converters based not only on quantum gates, such as [3] ($O(N)$), but also on quantum adders, such as [6,36], in terms of answer delay.

6 Conclusions

In this paper, a reversible circuit which is capable of computing the conversion from signed numbers to two's complement has been presented. This converter is an adaptation of a reversible out-of-place carry look-ahead adder presented in [36], simplifying it to the specific operation $\overline{a}+1$ (being a the number to be converted). We have carried out a deep analysis of the existing converter and adder circuits (current circuits that allow conversion from signed numbers to two's complement) to find the best ones considering a solid metric. We have justified that [6,36] are the best adders in terms of delay and quantum cost comparing them with the most modern circuits.

Once the best circuits have been identified, we have used them to design our proposal and, later, to compare the obtained results by all the studied circuits. Obtained results have shown that our proposed circuit outperforms the existing ones in terms of delay. The circuit improves the delay of [36] since all the Peres gates have been removed and several Toffoli gates have been replaced or simplified. Moreover, the number of normal input qubits has been reduced by half since the operations associated with the deleted inputs have been simplified.

An additional advantage of our proposal is that it does not contain any garbage output; therefore, the circuit could be entangled with any other reversible circuit which needs to operate with two's complement. As future work, we are planning to implement a new carry look-ahead converter to compute the two's complement of a number.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science throughout Project RTI2018-095993-BI00, by J. Andalucía through Project P12-TIC301 and by the European Regional Development Fund (ERDF). F. Orts is supported by an FPI Fellowship (attached to Project TIN2015-66680-C2-1-R) from the Spanish Ministry of Education. The authors wish to thank N.C. Cruz for his valuable support.

References

1. Baugh, C.R., Wooley, B.A.: A two's complement parallel array multiplication algorithm. IEEE Trans. Comput. **100**(12), 1045–1047 (1973)
2. Chaudhuri, A., Sultana, M., Sengupta, D., Chaudhuri, A.: A novel reversible two's complement gate (TCG) and its quantum mapping. In: Devices for Integrated Circuit (DevIC), 2017, pp. 252–256. IEEE (2017)
3. Chaudhuri, A., Sultana, M., Sengupta, D., Chaudhuri, C., Chaudhuri, A.: A reversible approach to two's complement addition using a novel reversible TCG gate and its 4 dot 2 electron QCA architecture. Microsyst. Technol. **25**(5), 1965–1975 (2019)
4. Cho, H., Swartzlander Jr., E.E.: Adder and multiplier design in quantum-dot cellular automata. IEEE Trans. Comput. **58**(6), 721–727 (2009)
5. Cuccaro, S.A., Draper, T.G., Kutin, S.A., Moulton, D.P.: A new quantum ripple-carry addition circuit (2004). arXiv preprint arXiv:quant-ph/0410184
6. Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry look-ahead adder (2004). arXiv preprint arXiv:quant-ph/0406142
7. Gupta, A., Singla, P., Gupta, J., Maheshwari, N.: An improved structure of reversible adder and subtractor (2013). arXiv preprint arXiv:1306.1889
8. Haghparast, M., Jassbi, S.J., Navi, K., Hashemipour, O.: Design of a novel reversible multiplier circuit using hng gate in nanotechnology. In: World Appl. Sci. J. Citeseer (2008)

9. Hung, W.N., Song, X., Yang, G., Yang, J., Perkowski, M.: Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(9), 1652–1663 (2006)
10. Islam, M.S., Rahman, M.M., Begum, Z., Hafiz, M.Z.: Fault tolerant reversible logic synthesis: carry look-ahead and carry-skip adders. In: International Conference on Advances in Computational Tools for Engineering Applications, 2009 (ACTEA'09). pp. 396–401. IEEE (2009)
11. Khosropour, A., Aghababa, H., Forouzandeh, B.: Quantum division circuit based on restoring division algorithm. In: 2011 Eighth International Conference on Information Technology: New Generations, pp. 1037–1040. IEEE (2011)
12. Kianpour, M., Sabbaghi-Nadooshan, R.: Novel 8-bit reversible full adder/subtractor using a QCA reversible gate. J. Comput. Electron. **16**(2), 459–472 (2017)
13. Koren, I.: Computer Arithmetic Algorithms. AK Peters/CRC Press, Boca Raton (2001)
14. Lanyon, B.P., Barbieri, M., Almeida, M.P., Jennewein, T., Ralph, T.C., Resch, K.J., Pryde, G.J., O'Brien, J.L., Gilchrist, A., White, A.G.: Simplifying quantum logic using higher-dimensional Hilbert spaces. Nat. Phys. **5**(2), 134–140 (2009)
15. Lemr, K., Bartkiewicz, K., Cernoch, A., Duek, M., Soubusta, J.: Experimental implementation of optimal linear-optical controlled-unitary gates. Phys. Rev. Lett. **114**(15), 153602 (2015)
16. Li, R., Alvarez-Rodriguez, U., Lamata, L., Solano, E.: Approximate quantum adders with genetic algorithms: an IBM quantum experience. Quantum Meas. Quantum Metrol. **4**(1), 1–7 (2017)
17. Ling, H.: High-speed binary adder. IBM J. Res. Dev. **25**(3), 156–166 (1981)
18. Lisa, N.J., Babu, H.M.H.: Design of a compact reversible carry look-ahead adder using dynamic programming. In: 2015 28th International Conference on VLSI Design (VLSID), pp. 238–243. IEEE (2015)
19. Markov, I.L., Saeedi, M.: Constant-optimized quantum circuits for modular multiplication and exponentiation (2012). arXiv preprint arXiv:1202.6614
20. Mazumder, M.: Synthesis of quantum circuit for full adder using khan gate. Int. J. Appl. Inn. Eng. Manag. (IJAIEM) **6**(6), 226–232 (2017)
21. Moghimi, S., Reshadinezhad, M.R.: A novel 4×4 universal reversible gate as a cost efficient full adder/subtractor in terms of reversible and quantum metrics. Int. J. Mod. Educ. Comput. Sci. **7**(11), 28–34 (2015)
22. Mohammadi, M., Eshghi, M.: On figures of merit in reversible and quantum logic designs. Quantum Inf. Process. **8**(4), 297–318 (2009)
23. Mokhtari, D., Rezai, A., Rashidi, H., Rabiei, F., Emadi, S., Karimi, A.: Design of novel efficient full adder architecture for quantum-dot cellular automata technology. Facta Universitatis, Series: Electronics and Energetics **31**(2), 279–285 (2018)
24. Montaser, R., Younes, A., Abdel-Aty, M.: New design of reversible full adder/subtractor using r gate (2017). arXiv preprint arXiv:1708.00306
25. Murali, K., Sinha, N., Mahesh, T., Levitt, M.H., Ramanathan, K., Kumar, A.: Quantum-information processing by nuclear magnetic resonance: experimental implementation of half-adder and subtractor operations using an oriented spin-7/2 system. Phys. Rev. A **66**(2), 022313 (2002)
26. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information, 10th edn. Cambridge University Press, Cambridge (2017)
27. Orts, F., Ortega, G., Garzón, E.M.: A quantum circuit for solving divisions using Grover's search algorithm. In: Proceedings of 18th International Conference on Computational and Mathematical Methods in Science and Engineering (2018)
28. Orts, F., Ortega, G., Garzón, E.M.: A faster half subtractor circuit using reversible quantum gates. Baltic J. Mod. Comput. **7**(1), 99–111 (2019)
29. Rahmati, M., Houshmand, M., Kaffashian, M.H.: Novel designs of a carry/borrow look-ahead adder/subtractor using reversible gates. J. Comput. Electron. **16**(3), 856–866 (2017)
30. Shukla, V., Singh, O., Mishra, G., Tiwari, R.: Design of a 4-bit 2's complement reversible circuit for arithmetic logic unit applications. In: The International Conference on Communication, Computing and Information Technology (ICCCMIT), Special Issue of International Journal of Computer Applications, pp. 1–5 (2012)
31. Takahashi, Y., Kunihiro, N.: A linear-size quantum circuit for addition with no ancillary qubits. Quantum Inf. Comput. **5**(6), 440–448 (2005)
32. Takahashi, Y., Kunihiro, N.: A fast quantum circuit for addition with few qubits. Quantum Inf. Comput. **8**(6), 636–649 (2008)

33. Takahashi, Y., Tani, S., Kunihiro, N.: Quantum addition circuits and unbounded fan-out. Quantum Inf. Comput. **10**(9), 872–890 (2010)
34. Talib, G.H.B., El-Maleh, A.H., Sait, S.M.: Design of fault tolerant adders: a review. Arab. J. Sci. Eng. **43**(12), 6667–6692 (2018)
35. Thapliyal, H.: Mapping of subtractor and adder-subtractor circuits on reversible quantum gates. In: Transactions on Computational Science XXVII, pp. 10–34. Springer (2016)
36. Thapliyal, H., Jayashree, H., Nagamani, A., Arabnia, H.R.: Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder. In: Transactions on Computational Science XVII, pp. 73–97. Springer (2013)
37. Theresal, T., Sathish, K., Aswinkumar, R.: A new design of optical reversible adder and subtractor using mzi. Int. J. Sci. Res. Publ. (IJSRP) **5**(4) (2015)
38. Toffoli, T.: Reversible computing. In: International Colloquium on Automata, Languages, and Programming, pp. 632–644. Springer (1980)
39. Wang, F., Luo, M., Li, H., Qu, Z., Wang, X.: Improved quantum ripple-carry addition circuit. Sci. China Inf. Sci. **59**, 042406 (2016)
40. Wang, J., Choi, K.: A carry look-ahead adder designed by reversible logic. In: SoC Design Conference (ISOCC), 2014 International, pp. 216–217. IEEE (2014)
41. Williams, C.P.: Explorations in quantum computing. Springer, Berlin (2010)

2.1.2 Efficient reversible quantum design of sign-magnitude to two's complement converters

Title	Efficient reversible quantum design of sign-magnitude to two's complement converters		
Authors	*F. Orts, G. Ortega and E.M. Garzón*		
Journal	Quantum Information and Computation		
Year	2020		
Volume	20(9 & 10)		
Pages	747-765		
DOI	https://doi.org/10.26421/QIC20.9-10-3		
IF (JCR 2020)	0.976		
Categories	Computer Science, Theory & Methods:	82/110	(Q3)
	Physics, Mathematical:	42/55	(Q4)
	Quantum Science & Technology:	15/17	(Q4)
	Physics, Particles & Fields:	26/29	(Q4)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

Quantum Information and Computation, Vol. 20, No. 9&10 (2020) 0747–0765
© Rinton Press

EFFICIENT REVERSIBLE QUANTUM DESIGN OF SIGN-MAGNITUDE TO TWO'S COMPLEMENT CONVERTERS

F. ORTS, G. ORTEGA, E.M. GARZÓN
University of Almería, Department of Informatics
Almería, 04120, Spain [a]

Received December 5, 2019
Revised July 20, 2020

Despite the great interest that the scientific community has in quantum computing, the scarcity and high cost of resources prevent to advance in this field. Specifically, qubits are very expensive to build, causing the few available quantum computers are tremendously limited in their number of qubits and delaying their progress. This work presents new reversible circuits that optimize the necessary resources for the conversion of a sign binary number into two's complement of N digits. The benefits of our work are two: on the one hand, the proposed two's complement converters are fault tolerant circuits and also are more efficient in terms of resources (essentially, quantum cost, number of qubits, and T-count) than the described in the literature. On the other hand, valuable information about available converters and, what is more, quantum adders, is summarized in tables for interested researchers. The converters have been measured using robust metrics and have been compared with the state-of-the-art circuits. The code to build them in a real quantum computer is given.

Keywords: Quantum Computing, Quantum circuits, Reversible circuit, Two's complement, Sign-magnitude representation to two's complement converter
Communicated by: R Jozsa & M Mosca

1 Introduction

Quantum computing uses the principles of quantum mechanics to define a new paradigm of computing that allows reaching goals that cannot be achieved by those already known algorithms or classical computing. There is a great interest in the scientific community for quantum computing [1]. Specifically, efforts must be made to determine what kind of problems can be solved more efficiently by quantum computing than by traditional computation. In any case, there are problems that can be solved using quantum computing more efficiently. Grover's and Shor's algorithms are the best examples of such problems [2, 3].

The most remarkable difference between both paradigms, quantum computing and classical computing, is the basic unit of each one. In the case of classical computing, the bit is the fundamental unit. A bit has only two possibilities, 0 and 1. Despite the simplicity of the bit, current computers can perform any type of operation if they have enough number of bits. On the other hand, quantum computers define their own unit, the quantum bit (qubit). These qubits can also be in states 0 or 1 and, what is more, because of the properties of quantum mechanics they can be simultaneously in both states. This property is called superposition.

[a]Corresponding author: F. Orts. e-mail: francisco.orts@ual.es.

Thanks to the superposition, a qubit can easily and completely imitate the behavior of, for example, an atom. For this reason among others, quantum computing is interesting in many research fields such as chemistry [4], image processing [5] or mathematics [6].

Nowadays, quantum computing is focused on making possible the implementation of the aforementioned Shor's algorithm. This algorithm allows a number N to be decomposed into factors. It is faster than any similar algorithm in classical computation [3]. The problem of computing Shor's algorithm does not come from the algorithm itself, but from the current quantum computers since they have only a few qubits [1]. However, companies like Google, IBM or Intel are currently working on getting larger quantum computers, that is, with more qubits. Meanwhile, to accelerate the achievements of the computation of Shor's algorithm, current works are focusing on optimizing the necessary resources for its computation.

Following the line of optimizing resources, it is especially important to optimize basic arithmetic operations. There are current works that present optimized versions to perform addition [7, 8], subtraction [9, 10, 11], multiplication [12, 13], and division [14, 15] in quantum computers. Very relevant is the addition, which has a fundamental role in Shor's algorithm. However, the optimization of the involved operations is not the only possibility to achieve the effective computation of Shor's algorithm and the advance of quantum computing in general. For example, another possibility is to take an appropriate numeric representation and to provide tools for its use. In classical computing, the use of the two's complement representation is common. Among other improvements, the two's complement allows the addition of negative numbers to be computed more efficiently [16]. Working in two's complement can be beneficial for the optimal use of the available resources in quantum computing, and for the acceleration of several operations [17, 18, 19, 20, 21, 22, 23, 24, 25].

Quantum circuits have several parameters such as the number of required qubits or the quantum cost (the number of used logical gates). This work proposes sign-magnitude to two's complement converters which are focused on optimizing resources to make their implementation as simple and inexpensive as possible. Circuits that involve a small number of resources are really appreciated in quantum computation. They are relevant even when they do not achieve quantum advantages since they may be a useful part in bigger circuits and algorithms [26]. To achieve this, a study of the state-of-the-art circuits that allow the conversion of sign-magnitude to two's complement has been carried out. This study has analyzed the benefits of each circuit to be able to implement new ones reducing their quantum cost, the number of involved qubits, the number of T gates, and other interesting parameters in an effort to minimize the use of resources. A solid metric has been defined to evaluate these parameters in both the proposed converters and the state-of-the-art circuits.

This paper is organized as follows. Section 2 presents background information on quantum gates, defines the metrics in terms of quantum computing, explains the two's complement methodology and finally explores the state-of-the-art circuits which can compute a conversion from sign-magnitude to two's complement and their methodology. In Section 3 the designs of the proposed sign-magnitude to two's complement converters are presented and discussed in detail. Finally, the conclusions are exposed in Section 4.

The circuits have been tested in the real quantum computer *ibmq_ourense*. Our implemented circuits are freely available through the following website: https://github.com/2forts/QuantumConversor. The code is ready to be used in the IBM Quantum Experience

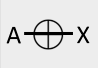
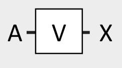
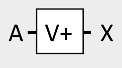
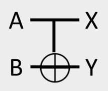
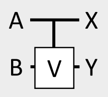
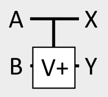
Name	**Pauli-X**	**V**	**V+**	**CNOT**	**Controlled-V**	**Controlled-V+**
Symbol						
Matrix Form	$\begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}$	$\begin{vmatrix} 1+i & 1-i \\ 1-i & 1+i \end{vmatrix}$	$\begin{vmatrix} 1-i & 1+i \\ 1+i & 1-i \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+i & 1-i \\ 0 & 0 & 1-i & 1+i \end{vmatrix}$	$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-i & 1+i \\ 0 & 0 & 1+i & 1-i \end{vmatrix}$

Fig. 1. Information of the basic reversible gates used in quantum circuits. (a) Pauli-X gate, (b) V gate, (c) V+ gate, (d) CNOT gate, (e) controlled-V gate, and (f) controlled-V+ gate.

platform, which includes quantum simulators and real quantum computers [27]. Due to the topology of each platform, several switch operations could be necessary. Luckily, the IBM Quantum Experience platform is able to perform them automatically.

2 Background

2.1 Measures in quantum circuits

This work has considered the metrics described in [28] to measure quantum circuits. Such metrics define five essential factors:

- Quantum cost: The quantum cost of a circuit defines the number of basic gates integrated into the circuit. This parameter is the most important when optimization of resources is mandatory [11].
- Delay: The delay of a circuit is its speed. A circuit is faster than other if its delay is lower.
- Normal inputs: inputs whose value is given by the user.
- Ancilla inputs: extra qubits used to compute auxiliary operations. Qubits remain a scarce resource, so the number of required qubits should be reduced whenever possible.
- Garbage outputs: outputs that are not part of the solution and whose value is unknown. For instance, an ancilla input whose value has not been uncomputed. All the outputs which are not part of the solution must be restored to its initial values or they will not be available to be entangled with other inputs of circuits since this operation will generate anomalous results [1].

In [28], authors set an important rule for measuring: the quantum cost of 1×1 and 2×2 gates is 1. Progressively, the quantum cost of an $N \times N$ gate consists of its number of 1×1 and 2×2 gates. The quantum cost of a circuit can be easily calculated adding the quantum cost of all its gates. Moreover, [28] defines the delay of 1×1 and 2×2 gates (that is, gates with quantum cost 1) as $1\triangle$. In the same way, the delay of an $N \times N$ gate is its depth when it

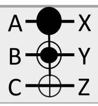
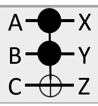
Name	Peres	Toffoli
Symbol		
Matrix Form	$\begin{vmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0 \end{vmatrix}$	$\begin{vmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{vmatrix}$

Fig. 2. Information of the basic reversible gates used in quantum circuits. (a) Peres gate and (b) Toffoli gate.

is built using 1×1 and 2×2 gates. In this work, all the measures of circuits have been carried out using these terms so that the comparisons were coherent. Therefore, the gates shown in Fig. 1 have a quantum cost of 1 and a delay of $1\triangle$. The Peres gate has a quantum cost of 4 and a delay of $4\triangle$, and the Toffoli gate a quantum cost and delay of 5 and $5\triangle$ respectively. Both gates are shown in Fig. 2.

Apart from the mentioned factors, because of quantum computers are extremely vulnerable to noise errors, other factors can be considered to include fault tolerance and palliate noise problems [29]. The T gates are used to make possible the use of error-correcting codes to ensure fault-tolerance in quantum circuits. But their drawback is that they are more expensive than the rest in terms of space and time cost due to their increased tolerance to noise errors [29, 30]. Therefore, there are two important factors to minimize the cost when using such gates: the T-count, that is, the number of T gates; and the T-depth, that is, the number of T gates which must be computed sequentially. Two or more T gates computed in parallel are at the same level and they only increase the T-depth in 1 unit. Therefore, the T-count and the T-depth of the circuits are also considered in this work.

2.2 Two's complement

Two's complement is a binary numeric representation. Its main advantage is that the representation of an integer is easily manipulated by the hardware. For instance, negating a number can be done inverting all its digits and also adding one to the whole number. Negation in sign-magnitude representation involves more operations because it is necessary to handle the sign of the number as a special digit. Other examples are the addition and subtraction. Adding negative numbers is identical to adding positive numbers. Therefore, it is not necessary additional logic to handle the negative case.

The representation of an N-digit number in two's complement consists of its complement with respect to 2^N. The numeric range of notation, for the N-digit case, is $-2^{N-1} \leq X \leq (-2^{N-1} - 1)$ [31]. As an example, the case $N = 4$ is shown in Table 1. Converting a number A from signed binary to two's complement is as follows. If $A >= 0$, no conversion is necessary as both representations are equal, that is, A has the same representation in signed binary as in two's complement. However, in the case of $A < 0$, the conversion is necessary. Such a conversion can be computed as the inversion of all the digits of A and then $\overline{\mathrm{A}} + 1$.

The conversion from sign-magnitude to two's complement can be done in two ways;

Table 1. Decimal, Sign-magnitude and Two's complement representation of binary numbers with $N = 4$.

Sign-magnitude	Two's complement	Decimal
0111	0111	7
0110	0110	6
0101	0101	5
0100	0100	4
0011	0011	3
0010	0010	2
0001	0001	1
0000	0000	+0
1000	–	−0
1001	1111	−1
1010	1110	−2
1011	1101	−3
1100	1100	−4
1101	1011	−5
1110	1010	−6
1111	1001	−7
–	1000	−8

- Designing a specific circuit which computes the conversion described in Table 1.
- Using an addition circuit.

Instead of using a specific converter, the conversion of a number A can be computed inverting its digits and then computing $\overline{\text{A}} + 1$. Therefore, the conversion can be done with N Pauli-X gates and an existing addition circuit. In the next subsection, the state-of-the-art converters and adders are reviewed.

2.3 Related work

Only a few works can be found in the literature related to the specific implementation of sign-magnitude to two's complement converters without using adders. And worse, none of

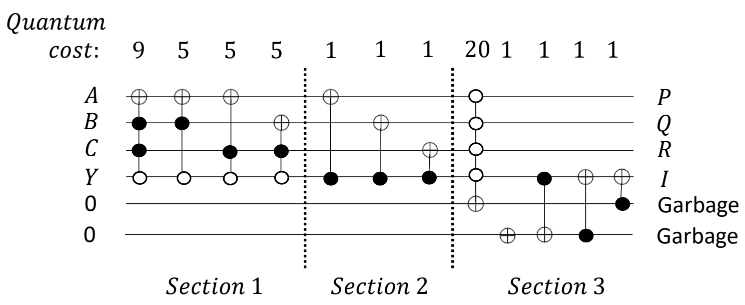


Fig. 3. Ripple-carry sign-magnitude to two's complement converter. It is based on the circuit presented in [32], adding it carry-in (Y) and carry-out (I). It accepts a number or a piece of the number of 3 digits (A, B, C), giving its conversion to two's complement (P, Q, R). It has two auxiliary inputs and two garbage outputs. Moreover, in the Figure can be observed the quantum cost of every step.

these works is based on fault-tolerant Clifford + T gates. [33] presented the mathematical design (but not a circuit design) of a gate called $SSMT$ which computes the conversion of a 4-digit number. [34] proposed a gate called TCG. This gate is only valid for the 4-digit case like the previous converter but, in this case, the logic design was presented. TCG was optimized in [32]. This design has a quantum cost of 25 for the case $N = 4$, being the best converter in terms of quantum cost. Although these converters have a good quantum cost, they are not able to compute numbers with more than 4 digits. They do not have carry input nor carry output, so several converters cannot be joined to compute numbers with more than 4 digits. In [35], a novel converter valid for any N-digit number was presented. It is based on a reversible out-of-place carry look-ahead adder presented in [8], adapted to compute the conversion specifically. Its delay improves the delay of the rest of converters and adders that are able to compute the conversion.

The sign-magnitude to two's complement converter of [32] has the lowest quantum cost, 35. However, as it has been mentioned, it is only valid for the case $N = 4$ as it does not support carry output nor carry input of any type. We have modified this gate with the objective to make it available for the general N-digit case, converting it in a ripple-carry converter. This circuit is shown in Fig. 3. Nevertheless, the cost of adding this extra feature is expensive because three qubits of the original gate need to be dedicated to the carry: on the one hand, it is necessary to add one new qubit to accept and compute the carry-in (Y in Fig. 3). We have replaced an original input qubit of the circuit of [32] by Y, so the circuit can compute a number (or a piece of the number) of $N = 3$. If $Y = 0$, the circuit acts like in [32] (Fig. 3, Section 1), but if $Y = 1$, the values of the inputs A, B and C are inverted (Fig. 3, Section 2). On the other hand, there are extra costs to generate the carry out (Fig. 3, Section 3). Two auxiliary qubits are needed. The first one is the target qubit of a multiple Toffoli gate (MTG, quantum cost of 20 [36, 37]), which is set to 1 for the case which generates carry-out, it means $\overline{A}\,\overline{B}\,\overline{C}\,\overline{Y}$. The second one is used to change the value of Y to I when necessary. The total quantum cost is 51, and the delay is also 51$\triangle$. The quantum cost of each gate of the circuit is shown in Fig. 3. However, the two garbage outputs have not been uncomputed, so the quantum cost and delay are assumed to be bigger than 51. For instance, following the Bennett's garbage removal scheme [38], it would be necessary four extra qubits to save the results, four extra CNOT gates to translate the result into these qubits, and also to apply the logical reverse of the entire circuit (except for the part which involves the new four qubits).

As it has been mentioned, the conversion of a number A can be done computing $\overline{\mathrm{A}} + 1$. For this, it is necessary to invert the digits of A using Pauli-X gates (one gate per digit) and to use an adder circuit to compute $\overline{A} + 1$. This is the most usual way to compute the two's complement since there are lots of quantum addition circuits in the state-of-the-art [7, 8, 11, 39, 40, 41, 42, 43, 44, 45], contrary to what happens with specific converters. Moreover, there are several fault tolerant adders [46, 47]. We have summarized the information of the most relevant adders in Table 2, but without including fault tolerant adders in this first comparison since they have a higher quantum cost due to their tolerance to noise errors. Adders with garbage outputs have not been included, for the reasons explained in subsection 2.1 about garbage outputs. Table 2 shows that the two most efficient addition circuits in terms of delay are [7, 8], with a delay of O(log N) (they are the only carry-lookahead adders in the table). However, they are expensive in terms of quantum cost. Ripple-carry adders improve them in terms of quantum cost. In general, ripple-carry adders are more optimized in terms of quantum cost, and carry look-ahead adder are better than them in terms of delay [48]. On the other hand, the ripple-carry adder which can be built using the full adder presented in [11] is the adder with the best quantum cost. They are adders with better quantum cost or delay, but they have garbage outputs [49, 50].

Table 2. Comparison of adders for N-digit numbers. The comparison only includes the ripple-carry and carry-lookahead adders which do not have garbage outputs and without the use of the fault-tolerant T gates.

Circuit	**Quantum cost**	**Delay** $\triangle$	**Ancilla inputs**	**Garbage outputs**
[7]	$28N - 15W(N) - 15log(N) - 6$	$logN + logN/3 + 7$	$5N/5$	0
[8]	$26N - 15W(N) - 15log(N-4)$	$logN + logN/3 + 2$	$5N/4$	0
[41]	$26N - 29$	$24N - 27$	0	0
[39]	$17N - 12$	$10N$	1	0
[51]	$15N$	$15N$	$3N$	0
[52]	$15N$	$10N$	N	0
[43]	$15N - 9$	$13N - 7$	0	0
[53]	$15N - 6$	$9N + 5$	0	0
[54]	$13N - 8$	$11N - 4$	0	0
[55]	$12N$	$12N$	$3N$	0
[56]	$12N$	$12N$	N	0
[56]	$12N$	$10N$	$4N$	0
[44]	$10N$	$8N$	N	0
[11]	$6N$	$4N$	N	0

The proposed circuits in this work are focused on optimizing the necessary resources of the current converters and adders acting as converters. Their details are presented in the next section.

3 Design of the new sign-magnitude to two's complement converters

We have developed three converters focused on minimizing the metrics of [28]. We propose three models:

- A first one focused on quantum simulators, which do not have noise problems. Its priorities are optimizing the quantum cost and the number of qubits.
- A second one focused on real quantum computers. Its priorities are achieving a balance between quantum cost, T-count and number of necessary qubits.
- A third one also focused on real quantum computes, but prioritizing T-count and number of necessary qubits.

3.1 First design: optimal quantum cost converter

We have developed a first sign-magnitude to two's complement converter focusing on the reduction of the quantum cost and the ancilla inputs. Neither the T-count nor the T-depth are taken into account since this design is focused on its use in simulators. This converter has two inputs: the digit to be converted (A) and the carry input (C_{in}). In the same way, it has

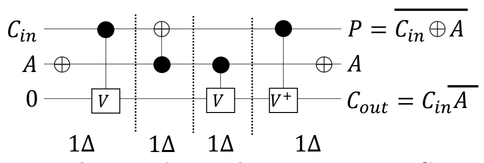

C_{in}
A
0
V
V
V^+
$P = \overline{C_{in} \oplus A}$
A
$C_{out} = C_{in}\overline{A}$
1Δ
1Δ
1Δ
1Δ

Fig. 4. Proposed sign-magnitude to two's complement converter. C_{in} is the input carry, A is the input, P is the output and C_{out} is the output carry. It has 1 auxiliary input and no garbage outputs. Its quantum cost is 6 and it has a delay of $4\triangle$.

two outputs: the converted digit (P) and the carry out (C_{out}). The idea behind this converter is that a sign-magnitude to two's complement conversion consists of negating a number and adding 1. That is, $\overline{\mathrm{A}} + 1$. Focusing on this specific addition, the conversion can be done digit by digit, taking into account that there is a carry which must be propagated. Since a digit only has two possibilities (0 or 1), its conversion depends on whether there is carry-in or not. Likewise, the conversion of such digit may generate carry-out or not. As there are only four possibilities (considering A and C_{in}), this can be expressed in a simple truth table as it is shown in Table 3.

Table 3. Truth table of the proposed converter.

C_{in}	A	P	C_{out}
0	0	1	0
0	1	0	0
1	0	0	1
1	1	1	0

C_{in} is the input carry, A is the input, P is the output and C_{out} is the output carry.

The proposed converter to compute the function of Table 3 has been built using Controlled-V, Controlled-V^+, $CNOT$ and Pauli-X gates. The circuit is shown in Fig. 4. It has 1 auxiliary input and no garbage outputs. It has been implemented using the properties of the Controlled-V and Controlled-V^+ gates described in Subsection 2.1. As the circuit has 1 V^+ gate and 2 V gates, the quantum cost of these gates is 3. It also uses a $CNOT$ gate, which has a quantum cost of 1, and two Pauli-X gates which also have a quantum cost of 1 each one. In conclusion, the circuit has a quantum cost of 6 and a delay of $4\triangle$. From Table 3 can be deduced that $P = \overline{C_{in}}\ \overline{A} + C_{in}A$ and $C_{out} = C_{in}\overline{A}$. On the one hand, $C_{in}\overline{A}$ can be computed using the properties of the Controlled-V^+ and Controlled-V gates. On the other hand, $\overline{C_{in}}\ \overline{A} + C_{in}A$ can be easily computed using a $CNOT$ gate. Although the $CNOT$ gate calculates the result inverted, the desired result can be achieved inverting A at the beginning with a Pauli-X gate. A is reverted at the end to avoid a garbage output.

For a better understanding of the circuit, all possible combinations and the obtained results are shown in Fig. 5. They coincide with the results of Table 3, verifying that the circuit works properly:

- Fig. 5(a) shows the case $C_{in} = 0$, $A = 0$. The second Controlled-V gate and the Controlled-V^+ are activated. $V(A) \times V^+(A) = V^+(A) \times V(A) = A$, so C_{out} maintains the initial value (0). The $CNOT$ gate is activated, so P is inverted to 1.

- Fig. 5(b) shows the case $C_{in} = 0$, $A = 1$. Neither gates Controlled-V nor Controlled-V^+

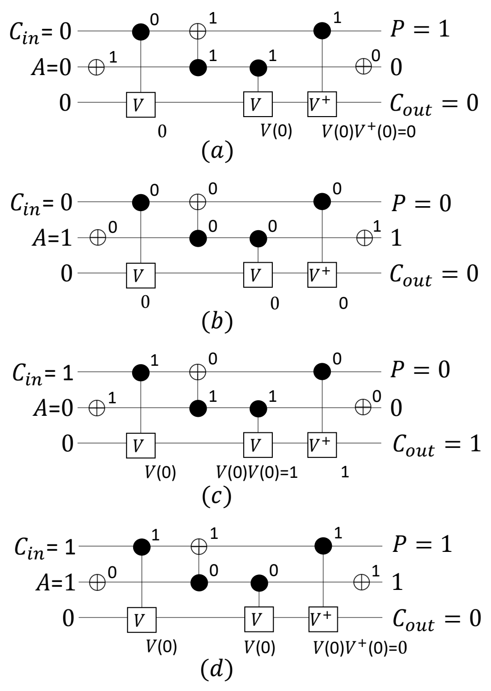


Fig. 5. For the sake of clarity, all the possible states of the proposed circuit are checked in detail, verifying that the obtained results are equal to those of Table 3. In (a), the circuit is tested for the case $C_{in} = 0, A = 0$. In (b), the circuit is tested for the case $C_{in} = 0, A = 1$. (c) shows the results of $C_{in} = 1, A = 0$. In (d), it is verified that the circuit also responds adequately for $C_{in} = 1, A = 1$.

are activated, so C_{out} maintains the initial value (0). The $CNOT$ gate is not activated, so P maintains its value.

- Fig. 5(c) shows the case $C_{in} = 1$, $A = 0$. The two Controlled-V gates are activated, so C_{out} inverts the initial value to 1. The $CNOT$ gate is activated, so P is inverted to 0.
- Fig. 5(d) shows the case $C_{in} = 1$, $A = 1$. The first Controlled-V gate and the Controlled-V^+ are activated. As $V(A) \times V^+(A) = V^+(A) \times V(A) = A$, C_{out} maintains the initial value (0). The $CNOT$ gate is not activated, so P maintains its value.

The real utility of the converter is its scalability, that is, the possibility of connecting it in cascade with other converters of the same type, to convert numbers of any size. A ripple-carry sign-magnitude to two's complement converter can be constructed with several converters in cascade, with the carry output from each converter connected to the carry input of the next one in the chain. In the general case, N converters can compute the conversion of an N-digit number (i.e. A), as it is shown in Fig. 6. In the figure, Converter 0 computes the least significant digit of A, it means A_0, and Converter $N-1$ computes the most significant digit of A (A_{N-1}). It is important to always start the process setting the initial carry input to 1. As an example, we explain how to convert the number -5 (1101 in sign-magnitude representation). The most significant digit is 1, so the value represented is negative. The

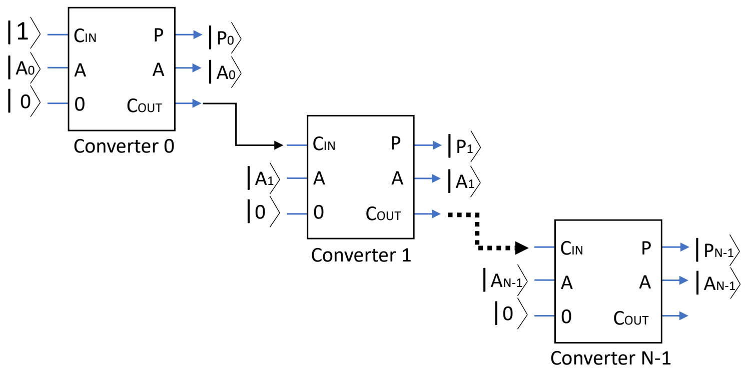


Fig. 6. Ripple-carry sign-magnitude to two's complement converters computing the conversion of an N-digit number.

number to be converted is 1101. Therefore, $C_{in0} = 1$ and $A = 101$ ($A_2 = 1, A_1 = 0, A_0 = 1$). Given that the number to be converted has 3 digits, the ripple-carry converter will consist of 3 converters:

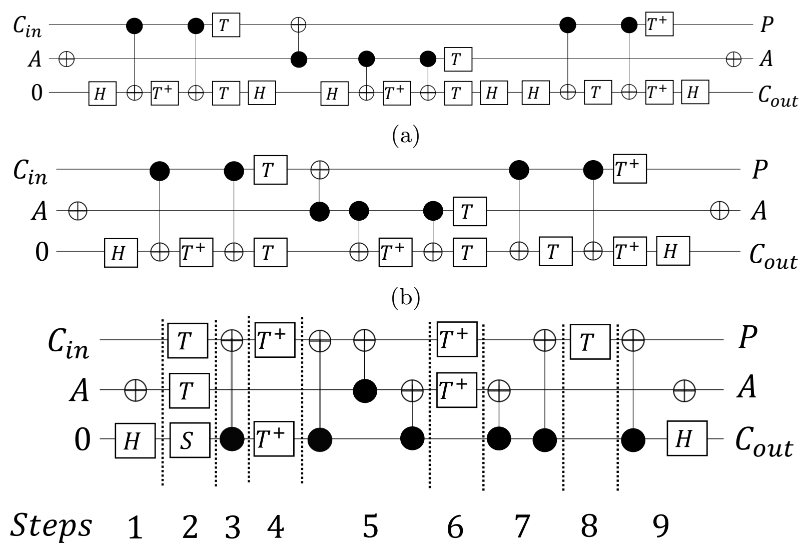
- The inputs of the Converter 0 are $C_{in0} = 1$ and $A_0 = 1$. According to Fig. 5(d), the results of this case are $P_0 = 1$ and $C_{out0} = 0$.
- The inputs of the Converter 1 are $C_{in1} = 0$ and $A_1 = 0$. According to Fig. 5(a), the results of this case are $P_1 = 1$ and $C_{out1} = 0$.
- The inputs of the Converter 2 are $C_{in2} = 0$ and $A_2 = 1$. According to Fig. 5(b), the results of this case are $P_2 = 0$ and $C_{out2} = 0$.

The result is given by P, that is, (1)011.

3.2 Second design: balance between quantum cost and T-count

A second circuit can be obtained from the circuit described in the previous subsection replacing its non-Clifford + T gates and optimizing the use of the T gates. It has the same number of qubits than the previous proposed converter and also no garbage outputs. The first step consists of achieving a circuit with only Clifford + T gates. In [58], a heuristic called "Initial expansion algorithm" is presented. This heuristic, which is very easy to be applied, is focused on transform a circuit into another circuit which only has Clifford + T gates. We have chosen the definition of the Controlled-V and V^+ in terms of Clifford + T gates proposed in [57] to expand these gates. The result is shown in Fig. 7a. Moreover, several Hadamard gates can be removed since they are applied consecutive, reducing the quantum cost of the circuit (Fig. 7b). Until now, we have limited ourselves to convert the original design to another one that uses only Clifford-T gates.

Finally, it is mandatory to optimize the T-count and the T-depth. The circuit can be optimized in these terms using a heuristic for the parallelization of the T gates [59]. Two T gates can be sequentially computed after several transforms, so they can be replaced by a Phase gate, reducing the T-count in one gate. At the end of the heuristic, the final circuit has a T-count of 8 and a T-depth of 4. The final circuit is shown in Fig. 7c. The quantum



(c)

Fig. 7: Second proposed converter. (a) Resulting circuit of expanding the Controlled-V and V^+ gates in the circuit of Fig. 4 as shown in [57] using the "Initial expansion algorithm" proposed in [58]. (b) Removing some redundant Hadamard gates from the previous circuit. (c) Minimization and parallelization of the T gates according to the method described in [59].

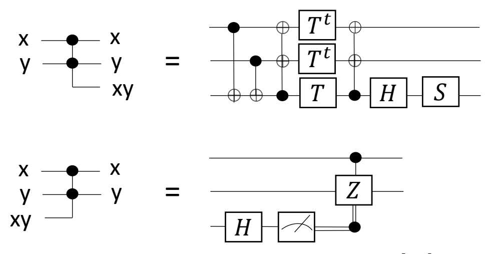


Fig. 8. Temporary logical-AND gate and its uncomputation gate [46]. The temporary logical-AND gate has a T-count of 4 and a T-depth of 2. The uncomputation gate does not involves T gates.

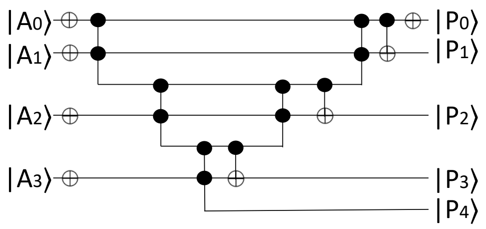


Fig. 9. Example of the third proposed converter, for the 4-digits case.

cost and delay of this new proposed circuit is bigger than the best ones presented in Table 5. However, to compare these circuits with the proposed one in this section in terms of quantum cost and delay is not coherent since the proposed circuit is the only one with fault tolerant capabilities.

3.3 Third design: optimal T-count and number of qubits

As it has been mentioned in Section 1, ripple-carry adders involve lesser resources than carry-lookahead adders. We can adapt the methodology of a ripple-carry adder, which computes $A + B$, to perform the operation $\overline{\mathrm{A}} + 1$. On the one hand, the resulting converter simplifies the operations of the original adder since $B_0 = 1$ and $B_i = 0$ for $i > 0$. For instance, The first carry out depends only on A_0: if A_0, then the carry out will be 0, and 1 otherwise as $A_0 + 1 = 10$ in this last case. The computation of the remaining carries also depends only of the digits of A. Therefore, the only difficulty is keeping a constant product of all the digits of A, which can be expensive if Toffoli gates are employed. On the other hand, all the qubits used to contain B can be removed, resulting in a significant reduction of necessary resources.

In [46], a new construction called temporary logical-AND gate, which performs the AND operation of two qubits into an ancilla qubit, is presented. This gate is similar to the Toffoli gate, but its T-count and T-depth are 4 and 2, respectively (whereas the T-count and T-depth of the Toffoli gate are 7 and 3, respectively. Moreover, its uncomputation does not need any T gate (the uncomputation of a Toffoli gate should be done using another Toffoli gate, so the same T-count and T-depth are required). We reproduce the temporary logical-AND gate and its uncomputation gate in Fig. 8. Using the reduced T-count and T-depth of this gate, we can achieve an optimized converter. An example of this converter for the 4-digit case is shown in Fig. 9. The circuit can be obtained for any N-digit number A following these steps:

1. For $i = 0$ to $i = N - 1$, apply a Pauli-X gate at every digit (qubit) A_i.

2. Apply a temporary logical-AND gate at A_0, A_1 over an ancilla qubit. We call this ancilla qubit as C_1, since its contains the output carry.

3. For $i = 2$ to $i = N - 1$, apply a temporary logical-AND gate at C_{i-1}, A_i over an ancilla qubit C_i.

4. For $i = N - 1$ to $i = 2$, apply a CNOT gate at C_{i-1} over A_i. Then, uncompute C_{i-1} using the uncomputation gate.

5. Apply a CNOT gate at A_0 over A_1, and finally a Pauli-X gate over A_0.

3.4 Cost analysis

The first proposed circuit in this work consists of 2 Controlled-V gates, 1 Controlled-V^+ gate, 1 $CNOT$ gate and 2 Pauli-X gates (quantum cost: 6). In the general case and assuming that N converters are necessary to compute the conversion of an N-digit number, the quantum cost is $6N$. In terms of delay and following the same metrics of [28], it has a delay of $4N\triangle$ since a gate with the quantum cost of 1 has a delay of $1\triangle$, and the Pauli-X gates can be computed in parallel with other gates, as it is shown in Fig. 4. Every converter has an auxiliary input, so the general case needs N auxiliary qubits. There are not garbage outputs in the proposed circuit. To calculate the T-count and T-depth of the Controlled-V gates we have considered the values given in [57]. Then, we conclude that the T-count and T-depth of the circuit are $9N$ and $6N$, respectively.

In this work, we have obtained the circuit directly from the truth table. However, it is also possible to use a SAT solver to quickly search the minimum quantum cost of a function with a small number of inputs [60]. To assure the quality of our circuit, we have checked it using such solvers. It can be concluded that the minimum cost of implementing the function given by the proposed circuit (considering the uncomputation of the garbage outputs) is 6, so we can affirm that the proposed circuit is optimal in that term. On the other hand, the number of garbage outputs cannot be improved since it is 0. The number of ancilla inputs (1) cannot be reduced either due to the maintenance of reversibility.

Focusing now on the second circuit, it has a T-count of 8 and a T-depth of 4, which can be obtained directly in Fig. 7c using the same metrics than in the previous circuit. The number of necessary qubits is $3N$ (the same number of the first circuit). It also has no garbage outputs. It optimizes the T-count and T-depth of the first circuit into $8N$ and $4N$, respectively.

Finally, focusing on the third circuit, only steps 2 and 3 require T gates as they involve temporary logical-AND gates. According to the T-count and the T-depth of the temporary logical-AND gate, it can be concluded that this circuit requires $4N-4$ T gates. These gates are computed sequentially, so the T-depth will be $2N-1$. The circuit requires N normal inputs (the digits of the number) and $N-1$ ancilla inputs, that is, a total of $2N-1$ qubits. For the rest of the metrics we have proceeded as in the previous cases, obtaining a quantum cost of $15-13$, a delay of $12N-10\triangle$, and 0 garbage outputs.

For the sake of clarity, we show the numbers of our three proposals in Table 4.

Table 4. Comparison of our three proposed converters for N-digit numbers, in terms of the metrics proposed in [28], but also in terms of T-count and T-depth.

Circuit	**Quantum Cost**	**Delay $\triangle$**	**Ancilla Inputs**	**Total Qubits**	**Garbage Outputs**	**T-Count**	**T-Depth**
First Proposed Circuit	$6N$	$4N$	N	$3N$	0	$9N$	$6N$
Second Proposed Circuit	$19N$	$13N$	N	$3N$	0	$8N$	$4N$
Third Proposed Circuit	$15N-13$	$12N-10$	$N-1$	$2N-1$	0	$4N-4$	$2N-2$

3.5 Cost comparison

3.5.1 Non Fault tolerant circuits

A comparison between several state-of-the-art converters and our first proposed circuit, in terms of quantum cost, delay, number of inputs, ancilla inputs and garbage outputs, is shown

Table 5. Comparison of converters for N-digit numbers.

Circuit	**Quantum cost**	**Delay** $\triangle$	**Normal inputs**	**Ancilla inputs**	**Garbage outputs**
[7]	$28N - 15w(N) - 15log(N) - 6$	$logN + logN/3 + 9$	$2N$	$5N/4$	0
[8]	$26N - 15w(N - 15log(N - 4)$	$logN + logN/3 + 4$	$2N$	$5N/4$	0
[35]	$21N - 15w(N) - 15log(N - 4)$	$logN + logN/3 + 1$	N	$5N/4 + 1$	0
[41]	$28N - 29$	$26N - 27$	$2N$	0	0
[39]	$19N - 12$	$10N + 2$	$2N$	1	0
[51]	$17N$	$15N + 2$	$3N$	$3N$	0
[52]	$17N$	$10N + 2$	$3N$	N	0
[43]	$17N - 9$	$13N - 5$	$2N$	0	0
[53]	$17N - 6$	$9N + 7$	$2N$	0	0
[54]	$15N - 8$	$11N - 2$	$2N$	0	0
[55]	$14N$	$12N + 2$	$2N$	$3N$	0
[56]	$14N$	$12N + 2$	$2N$	N	0
[56]	$14N$	$10N + 2$	$2N$	$4N$	0
[44]	$12N$	$8N + 2$	$3N$	N	0
[11]	$8N$	$4N + 2$	$3N$	N	0
[32] extended	$51(((N - 1)/3) + 1)$	$51(((N - 1)/3) + 1)$	$4((N - 1)/3 + 1)$	N	$2N$
Proposed Circuit 1	$6N$	$4N$	$2N$	N	0

The adders include two extra levels of delay (A level for the initial inversion and another to restore and avoid garbage outputs) and $2N$ extra quantum cost to act as a converters.

in Table 5. The converters presented in [32], [34] and [36] have not been included in the comparison since they are only valid for the $N = 3$ or $N = 4$ cases. We have also not included fault tolerant adders since these circuits will be compared to our fault tolerant proposals. Table 5 include all the adders described in Subsection 2.2, but acting as a converter in this case (this has extra quantum cost and delay derived to compute the inversion of all the digits of the number to be converted and to restore the number to avoid garbage outputs). [35] is a specifically designed circuit to compute the sign-magnitude to two's complement conversion. Notice that the circuits of Figs. 3 and 4 will hereinafter be referred to as [32] extended and proposed circuit, respectively.

Focusing on delay on Table 5, circuits [7], [8] and [35] are $O(logN)$, and the remaining circuits are $O(N)$. So, the fastest circuit of the comparison is [35] with a delay of $logN + logN/3 + 1\triangle$. Therefore, [35] is the best choice when speed is the priority. In terms of normal inputs (qubits which are not auxiliary), [11, 44, 51, 55] have $3N$ inputs (which are the number A to be converted), the 1 to be added to A (is 1 in the first digit and 0 in the remaining digits) and the carries. [32] extended has $4((N - 1)/3 + 1)$ inputs since it computes three digits (and the carry-in) at once. [7], [8] and the proposed circuit have $2N$ inputs. In the rest of the circuits except [35] and the proposed one, the inputs are A and also the 1 to be added. In the proposed circuit, the inputs are the digit and the carry-in. Finally, [35] optimizes the number of inputs since it was specifically designed to compute the conversion, so it only needs the number to be converted.

In terms of ancilla inputs, [7] and [8] have $5N/4$, and [35] has $5N/4 + 1$. These circuits prioritize the delay, using more qubits to achieve the highest speed. The most optimized circuits of the table have N ancilla qubits (they prioritize the reduction of the number of necessary qubits). In terms of garbage outputs, only [32] extended contains uncomputed outputs. It has been included only to demonstrate that an optimization starting from an existing specific converter was not possible (or, at least, not trivial). As it has been mentioned, the cost of adding carry-in and carry-out to the existing converters is expensive.

The quantum cost is the most important factor when the optimization of resources is the priority, especially considering the current scarcity of resources in quantum computing [1]. In terms of quantum cost, the proposed circuit is the best choice. It improves the quantum cost in a 33% with respect to [45] (which is not included in the table since it has garbage outputs) and [11], which are the circuits currently available with the best quantum cost. [44] has a quantum cost of $12N$, so the proposed circuit improves the quantum cost in a 50%. Circuits [7, 8, 35] have a higher quantum cost than the others since they decrease their delays at the expense of the quantum cost. Finally, [32] extended has an expensive quantum cost $(51(((N-1)/3)+1))$, as it has been mentioned. Focused on quantum cost, the proposed circuit overcomes the other options, therefore is the best option when the optimization of resources is mandatory.

Focusing on our proposal, we have demonstrated that our converter is the best circuit in terms of quantum cost. Moreover, Table 4 have shown that it outperforms the rest of the circuits in terms of delay, with the exception of the first three converters. However, these three circuits have a very expensive quantum cost, being non-viable in current quantum computers and simulators. In the same way, only the circuit of [35] improves our proposal in terms of normal inputs, but again at the cost of using three times more quantum gates than our proposal. In terms of ancilla inputs, several circuits outperform ours, but once again at the cost of using (at least) two times more gates. Our focus was to achieve a converter optimized in terms of quantum cost, but without neglecting the relevance of the rest of the metrics. Table 4 have shown that our proposal is the most balanced converter in all metrics.

3.5.2 Fault tolerant circuits

This comparison is focused on the T-count, T-depth and the number of required qubits to perform the operation. We mentioned that there are no fault tolerant converters in the literature. Fortunately, there are several fault tolerant adders. On the one hand, the adder presented in [46] is the best adder of the state-of-the-art in terms of T-count. On the other hand, [47] proposes four new adders. Since none of the adders included in [47] outperforms the adder of [46] in terms of T-count, we have selected the best of the four adders in terms of T-depth. Therefore, Table 6 shows a comparison between our converters and the best adders in terms of T-count, T-depth and number of qubits. Our third circuit outperforms the rest of the circuits in terms of T-count, with $4N-4$. The second best circuit in T-count is the circuit of [46], followed by our second proposal. The worst circuit in terms of T-count is [47], with a value of $16N - 8w(N) - 8log(N) - 4$ (where $w(n) = n - \sum_{i=1}^{\infty}(\frac{n}{2^y})$).

The circuit of [47] is the best option in terms of T-depth, with $4log(N) + 2log(\frac{2N}{3}) + 3$. The second best circuit is our third proposal, with $2N-2$, followed by [46] with a T-depth of $2N$, and by our second proposal with $4N$. Finally, in terms of number of qubits, the best option is again our third circuit, with only $2N-1$. The second best circuit is [46] with $3N-1$, and the third is our second proposal with $3N$. The worst is terms of necessary qubits is [47], which we have already seen that it sacrifices T-count and qubits to improve its T-depth.

To finish, we can conclude with the following classification:

- The best quantum cost: the first proposed circuit, with an improvement of a 33% with respect to the best current converters.

- The best delay: [35].

Table 6. Comparison of converters for N-digit numbers, in terms of T-count, T-depth and number of necessary qubits.

Circuit	**T-count**	**T-depth**	**Number of qubits**
Second Proposed Circuit	$8N$	$4N$	$3N$
[46]	$4N$	$2N$	$3N-1$
[47]	$16N - 8w(N) - 8log(N) - 4$	$4log(N) + 2log(\frac{2N}{3}) + 3$	$6N - 2w(N) - 2log(N)$
Third Proposed Circuit	$4N-4$	$2N-2$	$2N-1$

- Less number of inputs: [35] and the third proposed circuit.
- No garbage outputs: all the included except [32] extended.
- The best in terms of T-count: the third proposed circuit.
- The best in terms of T-depth: [47].
- Less number of qubits: The third proposed circuit.

4 Conclusion

In this work, we have presented the design of several scalable reversible sign-magnitude to two's complement converters. The first one optimizes the quantum cost, being also the most balanced in the rest of the metrics defined in [28]. Obtained results have shown that the quantum cost of our first proposed circuit improves a 33% with respect to the state-of-the-art circuits based on reversible gates and quantum adders. We have also presented a version of this circuit (the second proposal), achieving a balance between quantum cost and T-count. Additionally, we have designed an specific version (third proposal) that optimizes ancillary qubits and T-count. This last version outperforms the best fault tolerant circuits in the state-of-the-art. An advantage of our proposals is that they do not contain any garbage output, therefore they could be entangled with any other reversible circuit which needed to operate with two's complement. Moreover, we have demonstrated that adding carry-in and carry-out to the current best converter (in terms of quantum cost), which is limited to the case $N = 4$, is not a competitive option. A comparison, using a solid metric, between the proposed circuits and the best converters has been carried out in terms of quantum cost, delay, number of inputs, auxiliary qubits, garbage outputs, T-count, T-depth and number of qubits.

Additionally, we have analyzed the state-of-the-art converters and adders, giving valuable information summarized in tables which will be very useful for interested researchers in order to select the correct adder to their own applications.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science throughout Project RTI2018-095993-B-I00 and by the European Regional Development Fund (ERDF). F. Orts is supported by an FPI Fellowship (attached to Project TIN2015-66680-C2-1-R) from the Spanish Ministry of Education.

References

1. Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information 10th edition, 2017.

2. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.
3. Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
4. Hans Primas. *Chemistry, quantum mechanics and reductionism: perspectives in theoretical chemistry*, volume 24. Springer Science & Business Media, 2013.
5. Salvador Elías Venegas-Andraca. Introductory words: Special issue on quantum image processing published by quantum information processing. *Quantum Information Processing*, 14(5):1535–1537, 2015.
6. Elías F Combarro, José Ranilla, and Ignacio F Rúa. A quantum algorithm for the commutativity of finite dimensional algebras. *IEEE Access*, 7:45554–45562, 2019.
7. Thomas G Draper, Samuel A Kutin, Eric M Rains, and Krysta M Svore. A logarithmic-depth quantum carry look-ahead adder. *Quantum Information & Computation*, 6(4):351–369, 2006.
8. Himanshu Thapliyal, HV Jayashree, AN Nagamani, and Hamid R Arabnia. Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder. In *Transactions on Computational Science XVII*, pages 73–97. Springer, 2013.
9. KVRM Murali, Neeraj Sinha, TS Mahesh, Malcolm H Levitt, KV Ramanathan, and Anil Kumar. Quantum-information processing by nuclear magnetic resonance: Experimental implementation of half-adder and subtractor operations using an oriented spin-7/2 system. *Physical Review A*, 66(2):022313, 2002.
10. F Orts, G Ortega, and Ester M Garzón. A faster half subtractor circuit using reversible quantum gates. *Baltic Journal of Modern Computing*, 7(1):99–111, 2019.
11. Himanshu Thapliyal. Mapping of subtractor and adder-subtractor circuits on reversible quantum gates. In *Transactions on Computational Science XXVII*, pages 10–34. Springer, 2016.
12. Heumpil Cho and Earl E Swartzlander Jr. Adder and multiplier design in quantum-dot cellular automata. *IEEE Transactions on Computers*, 58(6):721–727, 2009.
13. Majid Haghparast, Somayyeh Jafarali Jassbi, Keivan Navi, and Omid Hashemipour. Design of a novel reversible multiplier circuit using hng gate in nanotechnology. In *World Appl. Sci. J.* Citeseer, 2008.
14. Alireza Khosropour, Hossein Aghababa, and Behjat Forouzandeh. Quantum division circuit based on restoring division algorithm. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 1037–1040. IEEE, 2011.
15. F Orts, G Ortega, and Ester M Garzón. A quantum circuit for solving divisions using Grover's search algorithm. In *Proc. 18th Int. Conf. Comput. Math. Method. Sci. Eng*, 2018.
16. Sarah Harris and David Harris. *Digital design and computer architecture: ARM edition.* Morgan Kaufmann, 2015.
17. Michael Kirkedal Thomsen, Robert Glück, and Holger Bock Axelsen. Reversible arithmetic logic unit for quantum arithmetic. *Journal of Physics A: Mathematical and Theoretical*, 43(38):382002, 2010.
18. Andreas Klappenecker and Martin Rotteler. Discrete cosine transforms on quantum computers. In *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.*, pages 464–468. IEEE, 2001.
19. Mihai Udrescu, Lucian Prodan, and Mircea Vlăduţiu. Implementing quantum genetic algorithms: a solution based on Grover's algorithm. In *Proceedings of the 3rd Conference on Computing Frontiers*, pages 71–82. ACM, 2006.
20. Fei Yan, Abdullah M Iliyasu, Yiming Guo, and Huamin Yang. Flexible representation and manipulation of audio signals on quantum computers. *Theoretical Computer Science*, 752:71–85, 2018.
21. S Manjula Gandhi, J Devishree, and S Sathish Mohan. A new reversible smg gate and its application for designing two's complement adder/subtractor with overflow detection logic for quantum computer-based systems. In *Computational Intelligence, Cyber Security and Computational Mod-*

els, pages 259–266. Springer, 2014.

22. Michael Nachtigal, Himanshu Thapliyal, and Nagarajan Ranganathan. Design of a reversible floating-point adder architecture. In *2011 11th IEEE International Conference on Nanotechnology*, pages 451–456. IEEE, 2011.
23. Ajinkya Borle and Samuel J Lomonaco. Analyzing the quantum annealing approach for solving linear least squares problems. In *International Workshop on Algorithms and Computation*, pages 289–301. Springer, 2019.
24. Maii T Emam and Layle AA Elsayed. Reversible full adder/subtractor. In *2010 XIth international workshop on symbolic and numerical methods, modeling and applications to circuit design (SM2ACD)*, pages 1–4. IEEE, 2010.
25. Jenil Jain and Rahul Agrawal. Design and development of efficient reversible floating point arithmetic unit. In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 811–815. IEEE, 2015.
26. Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.
27. Davide Castelvecchi. IBM's quantum cloud computer goes commercial. *Nature News*, 543(7644):159, 2017.
28. Majid Mohammadi and Mohammad Eshghi. On figures of merit in reversible and quantum logic designs. *Quantum Information Processing*, 8(4):297–318, 2009.
29. Edgard Muñoz-Coreas and Himanshu Thapliyal. T-count optimized design of quantum integer multiplication. *arXiv preprint arXiv:1706.05113*, 2017.
30. Fang Zhang and Jianxin Chen. Optimizing T gates in Clifford+ T circuit as pi/4 rotations around Paulis. *arXiv preprint arXiv:1903.12456*, 2019.
31. Israel Koren. *Computer arithmetic algorithms.* AK Peters/CRC Press, 2001.
32. Ayan Chaudhuri, Mahamuda Sultana, Diganta Sengupta, Chitrita Chaudhuri, and Atal Chaudhuri. A reversible approach to two's complement addition using a novel reversible tcg gate and its 4 dot 2 electron qca architecture. *Microsystem Technologies*, pages 1–11, 2018.
33. Vandana Shukla, OP Singh, GR Mishra, and RK Tiwari. Design of a 4-bit 2's complement reversible circuit for arithmetic logic unit applications. In *The International Conference on Communication, Computing and Information Technology (ICCCMIT), Special Issue of International Journal of Computer Applications*, pages 1–5, 2012.
34. Ayan Chaudhuri, Mahamuda Sultana, Diganta Sengupta, and Atal Chaudhuri. A novel reversible two's complement gate (tcg) and its quantum mapping. In *Devices for Integrated Circuit (DevIC), 2017*, pages 252–256. IEEE, 2017.
35. F. Orts, G. Ortega, and E. M. Garzón. An optimized quantum circuit for converting from sign–magnitude to two's complement. *Quantum Information Processing*, 18(11):332, Sep 2019.
36. D Michael Miller, Robert Wille, and Zahra Sasanian. Elementary quantum gate realizations for multiple-control toffoli gates. In *2011 41st IEEE International Symposium on Multiple-Valued Logic*, pages 288–293. IEEE, 2011.
37. Kamalika Datta, Indranil Sengupta, and Hafizur Rahaman. A post-synthesis optimization technique for reversible circuits exploiting negative control lines. *IEEE Transactions on Computers*, 64(4):1208–1214, 2014.
38. Charles H Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.
39. Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004.
40. Igor L Markov and Mehdi Saeedi. Constant-optimized quantum circuits for modular multiplication and exponentiation. *Quantum Information & Computation*, 12(5–6):361–394, 2012.
41. Yasuhiro Takahashi and Noboru Kunihiro. A linear-size quantum circuit for addition with no ancillary qubits. *Quantum Information & Computation*, 5(6):440–448, 2005.
42. Yasuhiro Takahashi and Noboru Kunihiro. A fast quantum circuit for addition with few qubits. *Quantum Information & Computation*, 8(6):636–649, 2008.

43. Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. *Quantum Information & Computation*, 10(9):872–890, 2010.
44. Feng Wang, Mingxing Luo, Huiran Li, Zhiguo Qu, and Xiaojun Wang. Improved quantum ripple-carry addition circuit. *Science China Information Sciences*, 59(4):042406, 2016.
45. Dmitri Maslov, Gerhard W Dueck, D Michael Miller, and Camille Negrevergne. Quantum circuit simplification and level compaction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):436–444, 2008.
46. Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, 2018.
47. Himanshu Thapliyal, Edgard Muñoz-Coreas, and Vladislav Khalus. T-count and qubit optimized quantum circuit designs of carry lookahead adder. *arXiv preprint arXiv:2004.01826*, 2020.
48. Md Saiful Islam, Muhammad Mahbubur Rahman, Zerina Begum, and Mohd Zulfiquar Hafiz. Fault tolerant reversible logic synthesis: Carry look-ahead and carry-skip adders. In *Advances in Computational Tools for Engineering Applications, 2009. ACTEA'09. International Conference on*, pages 396–401. IEEE, 2009.
49. Nusrat Jahan Lisa and Hafiz Md Hasan Babu. Design of a compact reversible carry look-ahead adder using dynamic programming. In *2015 28th International Conference on VLSI Design*, pages 238–243. IEEE, 2015.
50. Maryam Rahmati, Monireh Houshmand, and Masoud Houshmand Kaffashian. Novel designs of a carry/borrow look-ahead adder/subtractor using reversible gates. *Journal of Computational Electronics*, 16(3):856–866, 2017.
51. Shigeru Yamashita, Shin Minato, and D Michael Miller. DDMF: An efficient decision diagram structure for design verification of quantum circuits under a practical restriction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91(12):3793–3802, 2008.
52. Rajkumar Sarma and Ritika Jain. Quantum gate implementation of a novel reversible half adder and subtractor circuit. In *2018 International Conference on Intelligent Circuits and Systems (ICICS)*, pages 72–76. IEEE, 2018.
53. Himanshu Thapliyal and Nagarajan Ranganathan. A new reversible design of bcd adder. In *2011 Design, Automation & Test in Europe*, pages 1–4. IEEE, 2011.
54. Himanshu Thapliyal and Nagarajan Ranganathan. Design of efficient reversible logic-based binary and bcd adder circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 9(3):17, 2013.
55. William NN Hung, Xiaoyu Song, Guowu Yang, Jin Yang, and Marek Perkowski. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(9):1652–1663, 2006.
56. AN Nagamani, S Ashwin, and Vinod Kumar Agrawal. Design of optimized reversible binary adder/subtractor and BCD adder. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pages 774–779. IEEE, 2014.
57. Mathias Soeken, D Michael Miller, and Rolf Drechsler. Quantum circuits employing roots of the pauli matrices. *Physical Review A*, 88(4):042322, 2013.
58. D Michael Miller, Mathias Soeken, and Rolf Drechsler. Mapping NCV circuits to optimized Clifford+T circuits. In *International Conference on Reversible Computation*, pages 163–175. Springer, 2014.
59. Xianya He, Zhijin Guan, and Fei Ding. The mapping and optimization method of quantum circuits for Clifford+ T gate. *Journal of Applied Mathematics and Physics*, 7(11):2796–2810, 2019.
60. Daniel Große, Robert Wille, Gerhard W Dueck, and Rolf Drechsler. Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares. In *38th International Symposium on Multiple Valued Logic (ismvl 2008)*, pages 214–219. IEEE, 2008.

2.1.3 A review on reversible quantum adders

Title	A review on reversible quantum adders		
Authors	*F. Orts, G. Ortega, E.F. Combarro and E.M. Garzón*		
Journal	Journal of Network and Computer Applications		
Year	2020		
Volume	170 (102810)		
Pages	1-16		
DOI	https://doi.org/10.1016/j.jnca.2020.102810		
IF (JCR 2020)	6.281		
Categories	Computer Science, Hardware & Architecture:	5/53	**(Q1)**
	Computer Science, Interdisciplinary Applications:	14/112	(Q1)
	Computer Science, Software Engineering:	6/108	(Q1)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

Journal of Network and Computer Applications 170 (2020) 102810

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A review on reversible quantum adders

F. Orts [a,*], G. Ortega [a], E.F. Combarro [b], E.M. Garzón [a]

[a] *Group of Supercomputation-Algorithms, Dpt. of Informatics, Univ. of Almería, ceiA3, 04120, Almería, Spain*
[b] *Computer Science Department, Univ. of Oviedo, 33007, Oviedo, Spain*

ARTICLE INFO

Keywords:
Reversible adder
Quantum computing
Reversible circuit
Adder

2010 MSC:
00-01
99-00

ABSTRACT

Reversible adders are essential circuits in quantum computing systems. They are a fundamental part of the algorithms implemented for such systems, where Shor's celebrated factoring algorithm is one of the most prominent examples in which reversible arithmetic is needed. There is a wide variety of works in the existing literature which tackle the design of an adder for quantum systems, and today there is still a great interest in the creation of new designs and the perfection of the existing ones. Similar to how it happens in classical digital systems, there are different methodologies to approach the addition using reversible circuits. Some methodologies focus on minimizing the necessary resources, others on optimizing computing time, etc. In this work we analyze the reversible adders in the state-of-the-art for quantum computing, classifying them according to their type, and finally, comparing each other using referenced and validated metrics that allow highlighting the strengths and weaknesses of each adder.

1. Introduction

Reversible computation was first considered in the pioneering works of Landauer (1961), Lecerf (1963) and Bennett (1973) in the context of the energetic cost of computational operations. These authors unveiled deep connections between the thermodynamics of computation (in particular, the minimum amount of heat that a physical computing machine needs to dissipate per instruction) and the logical irreversibility of some operations. Surprisingly enough, it was discovered that the only computational task that implies an energy consumption is information erasure. Thus, in principle, computations may be physically executed without using energy as long as all operations are kept reversible and no information is lost in the process.

These profound results motivated further studies of, among others, Fredkin and Toffoli (Toffoli, 1980; Fredkin and Toffoli, 1982), who showed how any function computed by a logical circuit can be also computed by a reversible circuit. The key element is the existence of reversible gates that are universal in the sense that they can be used to simulate any other possible logic gate (reversible or not). With them, any circuit can be transformed in a reversible one with only a linear increase in the number of wires and gates (Fredkin and Toffoli, 1982). This opens the possibility of using reversible circuits in order to decrease the energetic consumption of computations, a topic that has gained interest in recent years (Cohen et al., 2016; Anamika, 2018; Chaves et al., 2018; Sahu et al., 2019).

Interest in reversible computation in general, and in reversible gates in particular, also comes from an intimate connection with quantum computing (Nielsen and Chuang, 2011). Quantum computing is a computational paradigm that exploits the physical properties of subatomic particles in order to achieve speedups in solving computational problems (Zhou et al., 2018; Zhang et al., 2018). Far from being solely a theoretical model, several quantum computer prototypes have been constructed in recent years (Linke et al., 2017; Michielse et al., 2017; Neill et al., 2018). In fact, Google has recently reported solving, with a quantum computer, a problem that would be unfeasible to solve with only classical resources, thus achieving the so-called quantum supremacy (Arute et al., 2019).

The main model of quantum computing is that of quantum circuits, in which logical gates are replaced with quantum ones (Nielsen and Chuang, 2011). These gates must obey the laws of quantum physics and, as a consequence, they are always reversible. Reversibility is therefore no longer the interesting energy saving option that we have described: it is now a fundamental requirement of quantum computing. This quantum paradigm requires us to reversibly implement even the most trivial algorithms that exist in classical computing. In fact, classical reversible gates such as the Toffoli gate play a very important role in quantum computing since they can be used, in combination with a few others, to approximate any possible quantum circuit (Shi, 2003). But in general,

* Corresponding author.
E-mail address: francisco.orts@ual.es (F. Orts).

https://doi.org/10.1016/j.jnca.2020.102810
Received 7 November 2019; Received in revised form 11 July 2020; Accepted 14 August 2020
Available online 28 August 2020

this conversion to a reversible methodology is not trivial, and implies an increase in the necessary resources compared to the classic counterpart (with the consequent effort to optimize their use), and even the search for more efficient alternative approaches from the point of view of reversibility (Vartiainen et al., 2004; Fowler et al., 2004).

Circuits for performing the addition are especially relevant for several quantum algorithms that achieve a speedup over the best known classical methods. Chief among them are Shor's algorithms, which can famously factor numbers and compute discrete logarithms in polynomial time (Shor, 1994), with momentous implications for classical cryptographic protocols such as the RSA cryptosystem (Rivest et al., 1978) or Diffie-Hellman key exchange (Diffie and Hellman, 1976). For instance, the most computationally intensive part of the algorithm for integer factorization is the modular exponentiation circuit. The most usual approach to compute the modular exponentiation is to use modular multiplier circuits, which are constructed using adders (Pavlidis and Gizopoulos, 2014). Although classically tractable, the design of the arithmetic part of the method usually requires considerable ingenuity in order to minimize the number of gates used and reduce the operational error, especially because all the operations must be conducted in a reversible way, making actual implementations highly non-trivial, as we have mentioned previously.

If checking that reversible adders are used in the most computationally critical part of the probably most important quantum algorithm were not enough to highlight the importance of such circuits, there are more examples. In addition to Shor's algorithms, quantum methods for achieving a quadratic speedup over classical algorithms in search and detection tasks have been proposed, most notably Grover's algorithm (Grover, 1996) and quantum walks (Venegas-Andraca, 2012). Although, in general, these methods do not involve arithmetical operations, they use a quantum oracle that is problem-depending and that may, in some cases, benefit from optimized reversible circuits for addition, as for instance when algebraic structures are involved (Combarro et al., 2019a; Combarro et al., 2019b).

Then, to build a circuit that implements any of these algorithms, is it necessary to design a reversible adder? Are there alternatives already implemented? In the literature, there is a wide variety of reversible circuits for basic arithmetic operations such as addition or multiplication. However, it is not always easy to analyze or compare them, because, on the one hand, the reported figures of merit are not consistent from one author to another and, on the other, not all parameters that are of potential interest are always clearly acknowledged. How can we know if a circuit is the right one for us if we do not have all of its information? How do we know that there is no better one?

For these reasons, together with the above mentioned connections of reversible circuits to computation energy reduction and to quantum computing, we think that a thorough, exhaustive, clear and impartial review of the existing reversible circuits for binary addition is needed. A review that seeks and establishes suitable metrics to accurately and verifiably measure a reversible circuit. A review that finds and analyzes the state-of-the-art adders based on these metrics, conveniently and visually offering all this information to anyone interested in using a reversible adder. A review that, in summary, is a reliable database of reversible adders. In this work, we aim to provide such a review, with special emphasis on being consistent on the parameters under which the circuits are evaluated and on highlighting their particular merits and flaws. We report the analysis of more than 40 references on reversible adders, clearly classifying them according to their different types and studying all their relevant parameters (including delay, quantum cost and the presence of garbage outputs). We also summarize all the pertinent information in several tables that interested researchers can quickly refer to in order to select the adder that is more suitable for their needs and provide original figures that exemplify some of the most prominent reversible adders for some values of their inputs.

The rest of the paper is organized as follows. In Section 2, we introduce and explain the different metrics that will be used to compare all the reversible adders studied in this work (Section 2.1), explaining how we have used them to carry out the review (Section 2.2). Since the adders are analyzed and compared based on their methodology, the types of adders and their main characteristics are presented in Section 3. Section 4 reviews the reversible adders that have been proposed in the literature, paying attention first to half-adders (Subsection 4.1), then to full adders (Subsection 4.2) and, finally, to carry propagate adders (Subsection 4.3, which includes ripple-carry adders and carry-lookahead adders). The comparison of all these adders is carried out in Section 5, where we also provide summary tables for quick reference of our findings in each category. Finally, in Section 6, we raise some conclusions of our study.

2. Metrics

2.1. Choice and justification of metrics

In the classical, non-reversible setting, measuring the complexity of a digital circuit is usually straightforward. A set of universal gates (for instance, AND, OR and NOT or just NAND) is fixed and the circuit complexity can be computed as the number of gates plus the number of bits that are needed to implement it together with a measure of its depth (which captures how many gates can be executed in parallel). When dealing with reversible circuits, in addition to considering the number of gates and the depth of the circuit, it is also important to take into account other aspects, such as the presence of garbage outputs. Also, as previously mentioned, one of the most important applications of reversible circuits comes from its use in quantum computing, something that affects the gates that can used to decompose the circuits. For these reasons, in this section we clearly define the parameters that will be used throughout the paper in order to study the complexity of reversible adders.

There are a large number of adder circuits available for quantum computing, as will be seen in this review. They all have a common goal: to make the addition of two numbers as efficient as possible. However, the concept of efficiency often changes among the authors of these circuits. And most importantly, each author frequently measures his circuit using the metrics he considers appropriate or even metrics defined by himself. Comparing adders becomes a tedious task because each circuit has been evaluated differently and therefore their metrics cannot be directly compared. We want to illustrate this problem with a specific example. Li et al. presented in Mohammadi et al. (2020) an adder which involves 28 quantum gates to perform and addition between two 5-digit binary numbers. On the other hand, Gidney presented an adder that needs 29 gates to perform the same operation (Gidney, 2018). However, none of them mentioned this information in their results. Li et al. evaluated their circuit in terms of the *quantum cost* and *delay*, while Gidney measured his circuit in terms of the *T-count*. This example reveals the difficulty in comparing the different quantum adders and the need to carry out a comparative study according to a wide and recognized set of characteristic parameters associated with quantum circuits.

The objective is to measure and to compare the existing adders using a common methodology that allows a direct comparison between them, also avoiding differences in the nomenclature. For instance, the quantum cost of a circuit is defined in several works as the number of gates which composes a circuit. According to this, a circuit which consists of 2 Toffoli gates has the same quantum cost than other circuit which consists of 2 CNOT gate s. Taking into account that a Toffoli gate is composed of 2 CNOT gates and other 3 gates (Nielsen and Chuang, 2011), this definition of quantum cost is imprecise. Moreover, an entire personalized circuit built with 5 Toffoli gates could be defined as a *novel reversible gate*, being its quantum cost 1. Comparing this new gate with a circuit which has 2 Toffoli gates would show that the first one has a quantum cost of 1 and the second one a quantum cost of 2.

This review is focused on the digital and logic levels of the adders. Therefore, exact and verifiable metrics at these levels are desirable. For these reasons, the metrics defined in Mohammadi et al. (2009) are followed. Four parameters are defined in Mohammadi et al. (2009) to evaluate reversible circuits:

- Quantum Cost (*QC*): the quantum cost of a circuit or a $X \times X$ gate is defined as the number of the 1×1 and 2×2 gates which composes it. The quantum cost of 1×1 and 2×2 gates is 1. This is a sensible metric, since we are mainly interested in the possibility of using arithmetical reversible circuits in quantum computing and most quantum computers use only 1×1 and 2×2 gates as primitives.
- Delay (*D*): the delay of a circuit defines its speed. A higher delay implies that a circuit is slower. $\triangle$ is the unit of delay defined in Mohammadi et al. (2009). 1×1 and 2×2 gates have a delay of $1\triangle$. The delay of a circuit or a $X \times X$ gate is defined by the number of 1×1 or 2×2 which must be computed sequentially. Therefore, if 2 or more gates can be computed in parallel, the delay will be determined by the delay of the slowest gate. To facilitate the evaluation of the delay, several schematic diagrams of this work graphically analyze the steps to complete the corresponding specific operations.
- Number of auxiliary Inputs (*I*): inputs which are set to a constant value (usually 0 or 1) and are used to do auxiliary operations.
- Garbage Outputs (*GO*): outputs which cannot be used at the end of the circuit since they have useless values. Garbage outputs must be reversibly removed (uncomputed) or these outputs may not be used later, which would result in a waste of resources. An output which is uncomputed to its original (and known) value is not considered as a garbage output. Uncomputing garbage outputs is especially important if the circuits are to be used in quantum computations, for garbage outputs can prevent the interference that quantum algorithms need to work properly.

According to Mohammadi et al. (2009), the quantum cost and delay of the basic gates used by the adders studied in this work are shown in Table 1, and their symbols in Fig. 1. Several gates are only used in specific adders, and they are analyzed along such adders.

The final idea is to show as much useful information as possible about a circuit (using the same metrics across all circuits to enable comparisons), understanding that there is no single better parameter. For example, on a machine with few resources, the general interest might be to reduce the number of qubits and the quantum cost; while in a machine that has more resources the interest could be to reduce the delay. However, we recognize that these metrics are not perfect and that it needs to be supplemented in some aspects. First, and as described below, there are various methodologies for performing addition. That is why we have considered it appropriate to classify and compare the adders according to their methodology instead of making a single comparison. The types of adders are explained in the next section. Second, there is a growing interest in implementing adders that allow the use of error detection and correction codes. These adders suffer from an increase in their metrics, but they have the advantage that they allow such error handling. In the review, we considered it convenient to indicate which adders have this capacity. It is important to mention that in these cases, the implementation of the gates in Table 1 may be different, increasing the quantum cost and the delay due to, usually, the incorporation of *T* gates. In relation to this matter, we also want to remark the work done in Gidney (2018), which is focused on improving the number of *T* gates needed to build *N*-bit adders.

Table 1
Gates and their quantum cost and delay.

Gate	QC	D
Pauli-*X*	1	1
V	1	1
V^+	1	1
Feynman/*CNOT*	1	1
Controlled-*V*	1	1
Controlled-V^+	1	1
Peres (Hung et al., 2006)	4	4
Toffoli (Nielsen and Chuang, 2011)	5	5

There is a third point to consider. As it has been mentioned, this work is focused on the logic level of the adders. However, it must be remarked that behind this level there are several physical realizations of these reversible gates and circuits like quantum computation, optic computation, quantum-dot cellular automata or ultra low power *VLSI* design (Thapliyal and Ranganathan, 2010). Each of these technologies has its own rules and limitations, which are out of the scope of this paper. For instance, we use the version of the Toffoli gate described in Nielsen and Chuang (2011) (except for several adders focused on error detection) since it optimizes the quantum cost and delay. Nevertheless, in linear optics, it is more important to optimize the number of controlled-unitary gates since the CNOT gate can only be probabilistically implemented (Orts et al., 2019). In these terms, versions of the Toffoli gate like the presented in Lanyon et al. (2009) and Lemr et al. (2015) are better options than the one described in Nielsen and Chuang (2011) since they are focused in reducing the number of controlled gates.

2.2. *Review methodology*

In this review, we have tried to analyze all the adders published at the time of writing these lines. However, it could be possible that we did not notice the existence of some adders due to the enormous amount of related works, which sometimes include the design of adders as part of a larger circuit without indicating it externally. Therefore, the existence of such adders goes unnoticed by anyone who does not read the article in depth. On the other hand, we have tried to make a thorough review in terms of the metrics described in the previous subsection. We have not limited ourselves to gather the information described in each work, but we have 1) implemented and tested the corresponding adder, and 2) measured the circuit using the proposed metrics.

For the implementation and testing of each adder, we have used the ProjectQ simulator, an open-source software framework for quantum computing (Steiger et al., 2018). The circuits have been implemented in Python under this framework and subjected to software tests to verify their correct operation. On the other hand, the measurement in terms of the metrics of Mohammadi et al. (2009) has also been done in Python over the circuits taking into account the following:

- The quantum cost can be easily measured setting a weight for each circuit and multiplying the number of gates of each type by its weight.
- The delay can be measured by dividing the circuit into levels in which no qubit acts twice. The delay of each level is given by the gate with the greatest weight.
- To count the number of ancilla inputs is trivial.
- The number of garbage outputs is measured by labeling the qubits which are not used to contain the result, and checking if they have been reverted symmetrically.

Some circuits offer designs adaptable to variable data size. In these cases, the circuit has been implemented in a way that dynamically adapts to the size of the input data. Thus it is possible to obtain the corresponding equation to each metric since the part to repeat of each circuit to increase it by each digit is perfectly defined.

Finally, we have made a comparison with the information obtained, gathering this information in tables to facilitate both its understanding and its use. In order not to make the comparison unnecessarily long, some of the analyzed adders have not been included. The main reason for discarding is the presence of garbage outputs in circuits that do not improve in any metric to those that do not present garbage outputs.

2.2.1. *About the implementation of functions*

Several types of adders are presented in the next section. The first

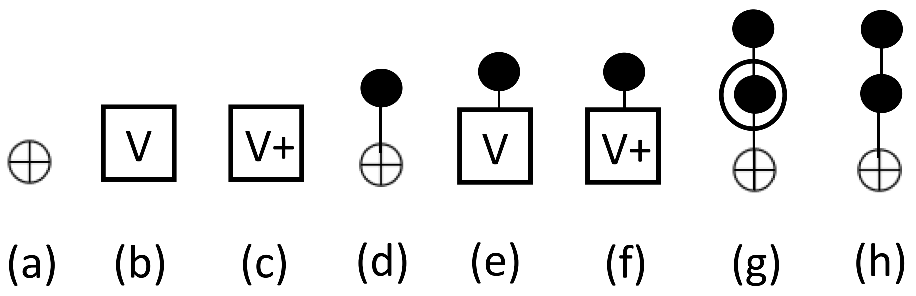


Fig. 1. Gates symbols: (a) Pauli-X, (b) V, (c) V^{+}, (d) Feynman/*CNOT*, (e) Controlled-V, (f) Controlled-V^{+}, (g) Peres and (h) Toffoli.

two, called *half adder* and *full adder*, are functions that implement a truth table of 2 and 3 inputs, respectively (see Tables 2 and 3). However, the implementation of these small functions has no merit: since we have the optimal implementations for the gates described in Table 1, it is possible to determine the optimal design of these functions in terms of the metrics described in this work using a SAT solver (Große et al., 2008). However, for the completeness of the review, we have included the analysis corresponding to the half and full adders.

2.2.2. *About error detection and correction codes*

We have mentioned that when analyzing an adder, we indicate whether or not it is fault-tolerant. However, an equivalent but fault-tolerant circuit can be obtained from any adder presented in this review by following these steps:

1. To apply the "Initial expansion algorithm" presented in Miller et al. (2014) to map the adder into a Clifford + T Circuit.
2. To remove redundant gates if necessary.
3. To minimize and parallelize the T gates according to the method described in He et al. (2019).

This review focuses on finding, analyzing, and comparing the work done by authors, so we do not make this adaptation in the circuits. Therefore, when in this review it is indicated that a circuit is fault-tolerant, it is because the authors of the adder have oriented its methodology to optimize it in these terms. In other words, authors present a circuit already prepared for fault-tolerance.

3. Types of adders

Addition is one of the basic operations in digital systems (Harris and Harris, 2015). Despite its apparent simplicity, there are a wide variety of ways to implement an adder. Since the review analyzes and catalogs the adders according to their type, it is important to make clear what each of the different types of adders consists of.

We have followed the terminology and classification order of adders described in Harris and Harris (2015) for classical adders:

- The *half adder* is the simplest case of an adder. This kind of circuit has two inputs: two digits A and B. Its objective is the computation of $A + B$. Notice that the result of the half adder needs two digits as the case $1 + 1$ returns 10. Therefore, the half adder has two outputs: S, which contains the least significant digit of the addition, and C_{out}, which contains the most significant digit (usually called carry out).

Table 2
Truth table of the half adder.

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 3
Truth table of the full adder.

C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 2 shows the truth table of the half adder. As consequence, it can be established that $C_{out} = AB$ and $S = A \oplus B$.

- A *full adder* is similar to a half adder, but accepting the carry in, C_{in}, as an input. Therefore, a full adder has 3 inputs (A, B and C_{in}) and 2 outputs S and C_{out}. According to its truth table (Table 3), it can be deduced that $S = A \oplus B \oplus C_{in}$ and $C_{out} = AB + AC_{in} + BC_{in}$.
- *Carry propagate adders* are able to sums two N-bit numbers A and B (usually with a carry in C_{in}). Their output consists on a N-bit number S, the result of the addition, and the carry out of that operation, C_{out}. The name *carry propagate adder* is used because the C_{out} of every pair of bits A_i and B_i is propagated into the next pair A_{i+1} and B_{i+1} (Harris and Harris, 2015). There are two kinds of carry propagate adders: ripple-carry adders and carry-lookahead adders.
  – A N-bit *ripple-carry adder* is built chaining N full adders, just connecting the C_{out} output of every full adder with the C_{in} input of the next full adder. This is shown in Fig. 2.
  – Carry-lookahead adders divide the addition into blocks to accelerate the computation of the carry out.

4. Analysis of adders

4.1. *Half adder*

On the one hand, a half adder can be built using a Toffoli gate to compute $C_{out} = AB$ followed by a CNOT gate to compute $S = A \oplus B$ (Nielsen and Chuang, 2011). This circuit has a quantum cost of 6, a delay of $6\triangle$, an auxiliary qubit and no garbage outputs, as it is shown in Fig. 3. This design has been widely used to implement schemes in different experimental systems (Chatterjee and Roy, 2015; Barbosa, 2006; Srivastava et al., 2017; Wu and Cain, 2014; Dridi et al., 2015; Eloie et al., 2018).

On the other hand, the Peres gate (Peres, 1985) can also act as a half adder (Akbar et al., 2011; Batish et al., 2018). The version of the Peres gate presented in Hung et al. (2006) achieves the best quantum cost among the 3×3 reversible gates (Thapliyal and Ranganathan, 2010). This version consists of 1 CNOT gate, 1 Controlled-V gate and 2 Controlled-V^{+} gates. The Peres gate has 3 inputs A, B and C, and produces 3 outputs $P = A$, $Q = A \oplus B$ and $R = AB \oplus C$. Setting $C = 0$, the outputs are $P = A$, $Q = A \oplus B$ and $R = AB$, which are the outputs of a half adder. Fig. 4 shows this use of the Peres gate. It has a quantum cost of 4, a delay

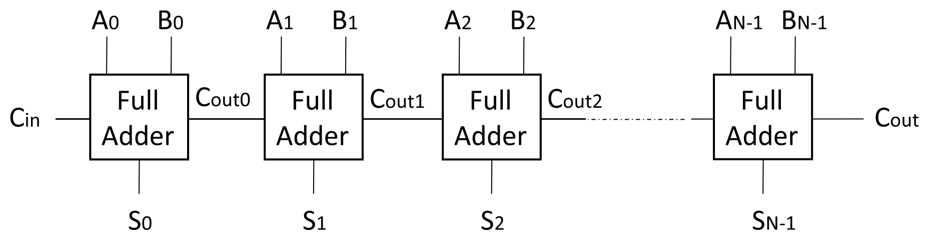


Fig. 2. *N* ripple-carry adder.

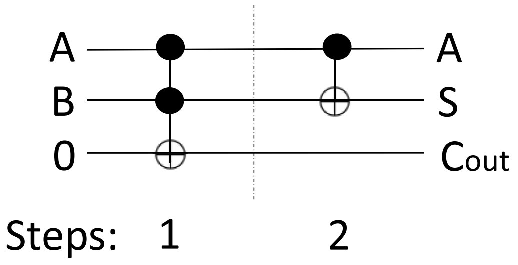


Fig. 3. Quantum implementation of the half adder proposed in Nielsen and Chuang (2011).

of 4△, an auxiliary qubit and no garbage outputs.

In Yamashita et al. (2008), a quantum circuit for half adder is mentioned in Fig. 3 of Section 2.2 as an example of semi-classical quantum circuit. The circuit is reproduced in Fig. 5. It has a quantum cost of 5 and a delay of 5△. To illustrate how it works, its truth table is shown in Table 4. In a similar way than the Peres gate, this circuit works as a half-adder if C is set to 0 and the variables P and Q denote C_{out} and S, respectively. Considering this case, the circuit has not garbage output and an auxiliary qubit.

There are several reversible half adder/subtractors (that is, circuits which compute half addition and subtraction at once) in the literature. These circuits have a quantum cost higher than regular half adders since they also perform the subtraction. A fault tolerant[1] full adder/subtractor using reversible gates was presented in Kaur and Dhaliwal (2012). The circuit of Kaur and Dhaliwal (2012) consists of two Feynman double gates (quantum cost 2 (Parhami, 2006)) and two Fredkin gates (quantum cost 5), with a total quantum cost of 14, the same delay, 3 auxiliary inputs, 1 selection qubit and 3 garbage outputs. Sarma and Jain (2018) presented a novel reversible half adder and subtractor circuit. This circuit has a quantum cost of 5, the same delay, 1 auxiliary qubit and no garbage outputs. It is shown in Fig. 6. Authors named this circuit *RSG* gate, and it can also be used to build a full adder/subtractor circuit. That functionality will be analyzed in the next section. In the same year, 2018, Balaji et al. (2018) presented a fault tolerant half adder/-subtractor, which is similar in term s of quantum cost to Kaur and Dhaliwal (2012) (it also consists of two Fredkin gates and two Feynman double gates). The circuit of Balaji et al. (2018) improves the number of garbage outputs and auxiliary inputs, from 5 to 3 and from 4 to 2 respectively. However, it has fan-out. Fan-out is not allowed in reversible logic design (Nielsen and Chuang, 2011). Both circuit s are shown in Fig. 7 and Fig. 8.

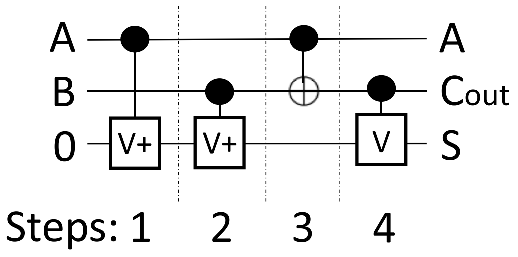


Fig. 4. Peres gate acting as a half adder, using the quantum implementation proposed in Hung et al. (2006).

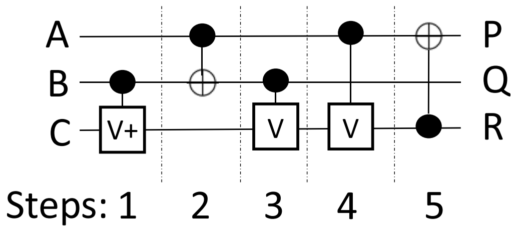


Fig. 5. Quantum implementation of the half adder presented in Yamashita et al. (2008).

Table 4
Truth table of the half adder of Fig. 5.

A	*B*	*C*	*P*	*Q*	*R*
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	0	0	1

4.2. Full adder

A simple way to build a full adder is to use three half-adders. A first half adder is used to compute $S_1 = A \oplus B$ and $C_{out1} = AB$. Then, a second half adder computes $S = S_1 \oplus C_{in}$ and $C_{out2} = S_1 C_{in}$. Finally, a third half adder computes $C_{out} = C_{out2} \oplus C_{out1}$ and an unused value (garbage)

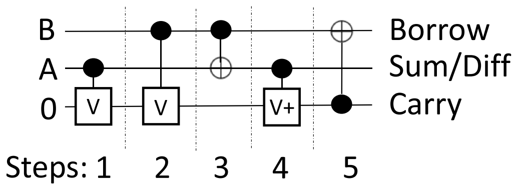


Fig. 6. Quantum implementation of the reversible half adder and subtractor circuit presented in Sarma and Jain (2018).

[1] A fault tolerant circuit protects the information while it dynamically undergoes computation. This kind of circuit is specially useful since the error probability per gate is guaranteed to be lower than a given constant threshold. Of course, they need extra quantum cost to achieve this result (Nielsen and Chuang, 2011). Although interesting, the study of the techniques used to achieve this remarkable result is beyond the scope of this review.

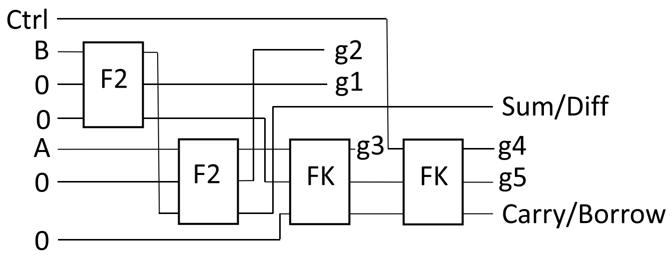


Fig. 7. Circuit of reversible fault tolerant half Adder/subtractor proposed in Kaur and Dhaliwal (2012). *F*2 represents a Feynman double gate, and *FK* a Fredkin gate.

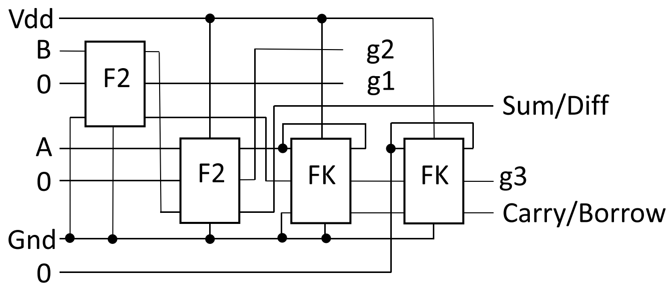


Fig. 8. Circuit of reversible fault tolerant half Adder/subtractor proposed in Balaji et al. (2018). *F*2 represents a Feynman double gate, and *FK* a Fredkin gate.

$C_{out2}C_{out1}$. This circuit is shown in Fig. 9, and can be built using any of the half adders described in the previous subsection. However, this design can be improved using 2 Peres gates (Bhagyalakshmi and Venkatesha, 2010). A first Peres gate computes $Q = A \oplus B$ and $R = AB$ (second and third outputs respectively), and a second one accepts Q, C_{in} and R as inputs to compute $S = Q \oplus C_{in}$ and $C_{out} = QC_{in} \oplus R$ (second and third outputs respectively). This circuit can be seen in Fig. 10. It has a quantum cost of 8, a delay of 8$\triangle$, 1 auxiliary qubit and 1 garbage output. These metrics have been calculated considering the version of the Peres gate presented by (Hung et al., 2006).

Khlopotine et al. (2002) proposed a full adder which uses the Fredkin gate (the Fredkin gate has a quantum cost of 5). This circuit consists of 5 Fredkin gates, so it has a quantum cost of 25. This version was improved in Bruce et al. (2002), reducing the necessary number of Fredkin gates into 4. In 2004, Cuccaro et al. (2004) proposed a new ripple-carry adder. It is based in 2 components (gates): a gate called Majority (*MAJ*), and another called UnMajority (*UMA*). These two gates can be combined to act as a full adder. The *MAJ* gate has three inputs, C_{in}, B and A, and three outputs, $U = C_{in} \oplus A$, $V = B \oplus A$ and C_{out}. Once the *MAJ* gate has been applied, C_{out} must be used or saved since the computation of the *UMA* gate will reverse this value into A. When the use of C_{out} is finished, the *UMA* gate is computed. It has three inputs (U, V and C_{out}) and three outputs: C_{in} and A (those values are reversed to avoid garbage outputs) and the sum S. The complete circuit to compute this process is shown in Fig. 11. It has a quantum cost of 14, the same delay, 0 auxiliary qubits and no garbage outputs (the complete ripple-carry adder of Cuccaro et al. (2004) will be analyzed in a later section). The design of Cuccaro et al. (2004) was improved in later works (Takahashi and Kunihiro, 2005; Takahashi and Kunihiro, 2008; Trisetyarso and Van Meter, 2010; Meter et al., 2008). In 2016, Wang et al. (2016) proposed a new design which keeps C_{out}. This circuit is shown in Fig. 12. It has a quantum cost of 10, a delay of 8$\triangle$, 1 auxiliary qubit and no garbage outputs.

Maslov et al. (2008) designed a full adder which consists of 1 Controlled-V^{+} gate, 3 Controlled-V gates and 2 CNOT gates. As it is shown in Fig. 13, this adder has a quantum cost of 6, a delay of 4$\triangle$, 1 auxiliary qubit and 1 garbage output. In Nagamani et al. (2014), it was presented a full adder with a quantum cost of 12, delay 12$\triangle$ and keeping 1 auxiliary qubit but avoiding garbage outputs. A circuit proposed in Thapliyal (2016) improves them. This circuit has the same quantum cost, delay and number of auxiliary qubits than (Maslov et al., 2008), but with no garbage outputs. It consists of 3 Controlled-V^{+} gates, 1 Controlled-V gate and 2 CNOT gates. The circuit of Thapliyal (2016) is shown in Fig. 14. Also in 2016, Singh and Rai (2016) proposed two alternative designs of full adder based on reversible gates, but none of them improves the adder of Thapliyal (2016). The best of the adders of Singh and Rai (2016) has a quantum cost of 8, a delay of 8$\triangle$, 1 auxiliary input and 1 garbage output.

Several fault tolerant full adders have been proposed. As it was mentioned, a fault tolerant circuit has a higher quantum cost because of parity preservation (Valinataj et al., 2016). For instance, Mitra and Chowdhury (2012) proposed a fault tolerant full adder with a quantum cost of 11, the same delay, 2 auxiliary inputs and 3 garbage outputs.

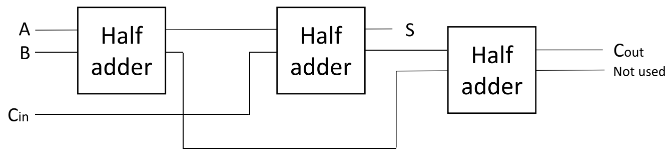


Fig. 9. Quantum implementation of a full adder using half adders.

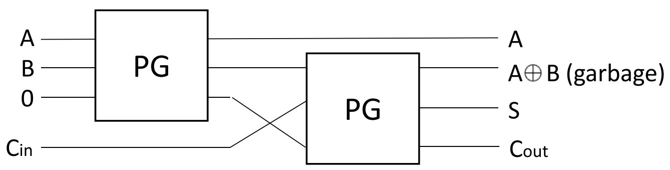


Fig. 10. Quantum implementation of a full adder using two Peres gates.

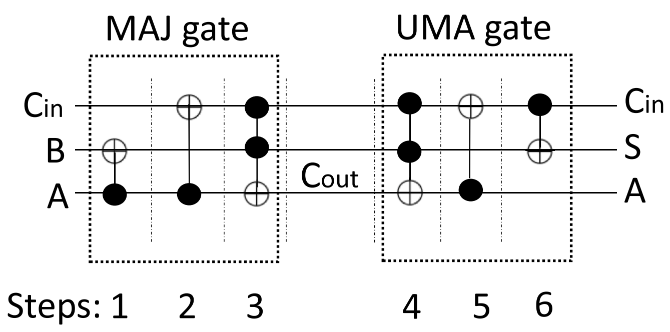


Fig. 11. Full adder proposed in Cuccaro et al. (2004). It consists of two gates called MAJ and UMA. C_{out} must be used before applying the UMA gate.

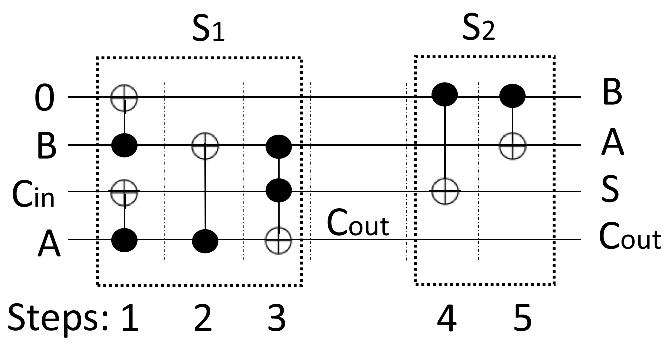


Fig. 12. Full adder proposed in Wang et al. (2016). The first sub-circuit $S1$ computes C_{out} and the second one computes S starting from the outputs of $S1$ without erasing C_{out}.

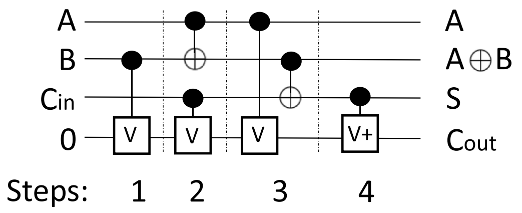


Fig. 13. Full adder proposed in Maslov et al. (2008).

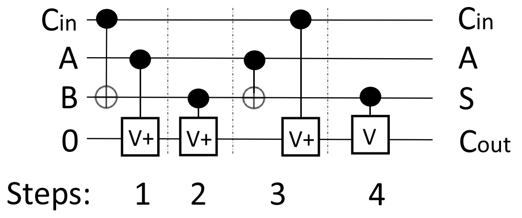


Fig. 14. Full adder proposed in Thapliyal (2016).

Previously to Mitra and Chowdhury (2012), several fault tolerant full adders were proposed: (Islam et al., 2009a) with a quantum cost of 14 and 3 garbage outputs, Bruce et al. (2002) which has been already analyzed in this section, and Haghparast and Navi (2008) with a quantum cost of 18 and 6 garbage outputs. Other fault tolerant adders are (Islam et al., 2009b) with a quantum cost of 14 and 3 garbage outputs, Dastan and Haghparast (2011) with a quantum cost of 14 and 3 garbage outputs, and Zhou et al. (2014) with a quantum cost of 8, delay 7 and 2 garbage outputs (Fig. 15). The circuit of Valinataj et al. (2016) has a quantum cost of 10 and 3 garbage outputs, but it offers interesting benefits against (Zhou et al., 2014) in terms of the transistor count or the total logical calculation (the number of *XOR*, *AND*, and *NOT* operations).

Similar to what happened with half adders, there are several reversible full adder/subtractors in the literature. Again, these circuits have a quantum cost higher than normal full adders since they also perform the subtraction. Rangaraju et al. (2010) proposed three designs. The best one consists of 2 CNOT gates and 2 Peres gate. It has a quantum cost of 10, the same delay, 1 auxiliary qubit and 3 garbage outputs. It also needs an extra selection qubit in order to select the operation to be computed (addition or subtraction). The half adder/subtractor of Kaur and Dhaliwal (2012) can be used to build a fault tolerant full adder/subtractor. 2 of these half adder/subtractors and 1 Feynman double gate (quantum cost 2 (Parhami, 2006)) are needed, with a total quantum cost of 30, the same delay, 9 auxiliary inputs, 1 selection qubit and 11 garbage outputs. A similar circuit was proposed in Saligram and Rakshith (2013), which consists of 4 Feynman double gates and 2 Fredkin gates, reducing the quantum cost to 18, the auxiliary inputs to 5 and the number of garbage outputs to 6. Moreover, Kumar et al. (2017) improved this design, using 3 Feynman double gates and only 1 Fredkin gate (total quantum cost of 11). It has 4 garbage outputs and 4 auxiliary inputs. The full adder of Thapliyal (2016) (Fig. 14) can be converted into a full adder/subtractor adding a selection qubit and a CNOT gate, having a quantum cost of 8 and a delay of 5△. On the other hand, the *RSG* gate presented in Sarma and Jain (2018), whose use as half adder/subtractor has been studied in the previous section, can be used to built a full adder/subtractor. It is able to compute both operations in parallel, without a selection qubit. This circuit has a quantum cost of 15, a delay of 10△, 1 auxiliary input and no garbage outputs. If we only consider the adder path of the circuit, the quantum cost is reduced to 10. Also in 2018, Balaji et al. (2018) proposed a fault tolerant full adder/subtractor using its half adder/subtractor described in the previous section. The complete circuit requires 2 of those half adder/subtractor s (quantum cost of 14 each one) and 1 Feynman double gate. Its final quantum cost is 30, with 5 auxiliary inputs and 3 garbage outputs.

In Batish et al. (2018), a comparative analysis for performance evaluation of reversible full adders is carried out. As a part of the analysis, they considered several methods to implement full adders using reversible gates:

- Full adder using *PCTG* gates: the *PCTG* gate consists of 1 Fredkin gate and 1 Feynman double gate. The full adder is built using 2 of these gates and 2 Feynman double gate s. As it is shown in Fig. 16, it has a quantum cost of 18, the same delay, 5 auxiliary inputs and 6 garbage outputs.
- Full adder using *BKG* gates: this gate was defined in Bhuvana and VS (2016). In that work, it is said that the *BKG* has a quantum cost of 1 since it is only 1 gate. However, according to the metrics of Mohammadi et al. (2009), a 4 × 4 gate cannot have a quantum cost of 1. The internal design of this gate is not described. It is detailed that it has four inputs *A*, *B*, *C* and *D*, and four outputs $P = A, Q = A\overline{D} \oplus C, R = (A\overline{D} \oplus C) \oplus B$ and $S = (A\overline{D} \oplus C)B \oplus AC \oplus A\overline{D}$. Setting $D = 0$, it acts as a full adder with 1 garbage output.
- Full adder using *DKG* gates: this circuit is defined in Krishnaveni et al. (2012). Similar to *BKG* gate, the internal design of this gate is not described. It has four inputs A', B', C' and D', and four outputs $P = B', Q = \overline{A'}C' + A'\overline{D'}, R = (A' \oplus B')(C' \oplus D') \oplus C'D'$ and $S = B' \oplus C' \oplus D'$. If the inputs were set to $A' = 0, B' = A, C' = B$ and $D' = C_{in}$, the outputs would be $P = A, Q = B, R = A(B \oplus C) \oplus BC = C_{out}$ and $S = A \oplus B \oplus C = Sum$. According to Mohammadi et al. (2009), it has no garbage outputs.
- Full adder using Peres gates: this can be seen in Fig. 10.
- Full adder using Peres and CNOT gates: this idea was introduced in Rohini and Rajashekar (2016). It s quantum cost is higher than the version of Fig. 10, and it has more garbage outputs.
- Full adder using *IG* gates: the *IG* gate was presented in Islam et al. (2009c). It has 4 inputs *A*, *B*, *C* and *D*, and four outputs $P = A, Q = A \oplus B, R = AB \oplus C$, and $S = BD \oplus \overline{B}(A \oplus D)$. Two *IG* gates connected in cascade can act as a full adder, as shown in Fig. 17. It has 3 garbage outputs. Once again, the internal design is not covered.

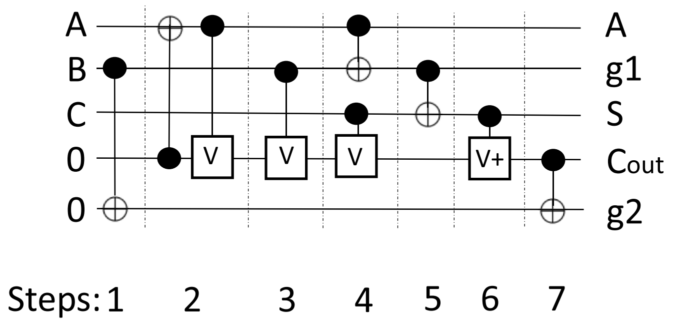


Fig. 15. Fault tolerant full adder proposed in Zhou et al. (2014).

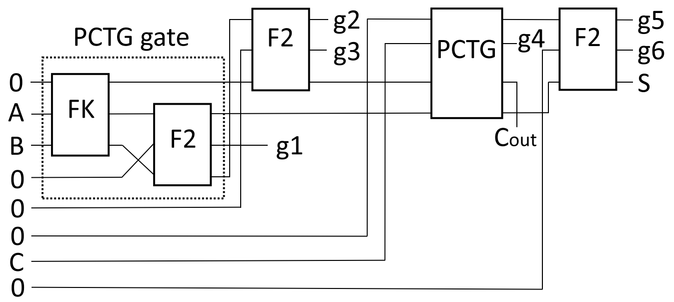


Fig. 16. Full adder using *PCTG* gates. A *PCTG* gate consists of 1 Fredkin gate (*FK*) and 1 Feynman double gate (*F2*).

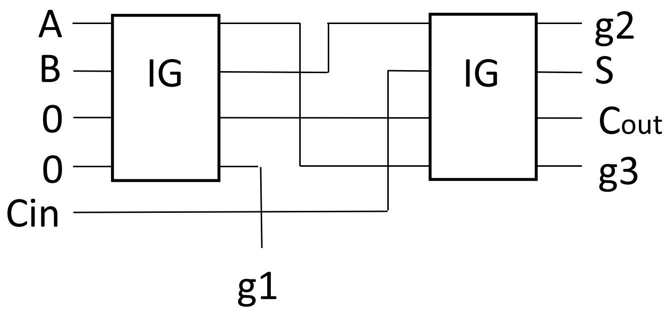


Fig. 17. Full adder using *IG* gates.

- Full Adder using Feynman and Fredkin gates: this full adder is the version proposed in Singh and Rai (2016), which has already been studied in this section.

4.3. Carry propagate adder

4.3.1. Ripple-carry adder

Since a *N*-digit ripple-carry adder is composed of *N* full adders, its *QC*, *D*, *I* and *GO* are given by the following equations:

$$\begin{aligned} QC_{ripple} &= N \cdot QC_{fulladder} \\ D_{ripple} &= N \cdot D_{fulladder} \\ I_{ripple} &= N \cdot I_{fulladder} \\ GO_{ripple} &= N \cdot GO_{fulladder} \end{aligned}$$

If the ripple-carry adder did not have carry in ($C_{in} = 0$), the least significant full adder could be replaced by a half adder. In this case, the equations are:

$$\begin{aligned} QC_{ripple} &= (N-1) \cdot QC_{fulladder} + QC_{halfadder} \\ D_{ripple} &= (N-1) \cdot D_{fulladder} + D_{halfadder} \\ I_{ripple} &= (N-1) \cdot I_{fulladder} + I_{halfadder} \\ GO_{ripple} &= (N-1) \cdot GO_{fulladder} + GO_{halfadder} \end{aligned}$$

It is possible to use any of the full adders of subsection 4.2 to build a ripple-carry adder. For the case $C_{in} = 0$, the least significant full adder can be replaced by any of the half adders of subsection 4.1. Ripple-carry adders are the best of the carry propagate adders in terms of quantum cost. However, due to their linear nature, they have a delay higher than others adders since the carry signals must propagate though every pair of bits A_i and B_i (Harris and Harris, 2015). In terms of quantum computers, which currently have few resources, this kind of adders are the best option as they minimize the quantum cost.

In addition to the ripple-carry adders that can be formed by combining the described full (and half) adders, it is worth noting some special cases. For instance, in Cuccaro et al. (2004) a ripple-carry adder is built using their full adder (Fig. 11). Then, this ripple-carry adder is optimized swapping and avoiding unnecessary gates. Assuming $C_{in} = 0$, the circuit can be even optimized further. For the general case, the circuit needs $2N - 1$ Toffoli gates, $5N - 3$ *CNOT* gates, and $2N - 4$ Pauli-*X* gates, with a total quantum cost of $(2N-1) \times 5 + (5N-3) \times 1 + (2N-4) \times 1 = 17N - 12$. The delay is $10N\triangle$ as it has $2N - 1$ Toffoli time-slices and 5 *CNOT* time-slices ($(2N-1) \times 5 + 5 \times 1$). It has 1 auxiliary input and no garbage outputs. As an example, the optimized circuit for the case $N = 6$ is shown in Fig. 18. Other proposals in the literature which presented a ripple-carry adder without C_{in} are (Takahashi and Kunihiro, 2005) (Quantum cost: $26N - 29$, delay: $24N - 27\triangle$, number of auxiliary inputs: 0, number of garbage outputs: 0), (Takahashi et al., 2010) (QC: $15N - 9$, D: $13N - 7\triangle$, AI: 0, GO: 0), and Thapliyal and Ranganathan (2013) (QC: $13N - 8$, D: $11N - 4\triangle$, AI: 0, GO: 0). The circuit of Thapliyal and Ranganathan (2013) is shown in Fig. 19 for the case $N = 4$.

On the other hand, there are several proposals (Cuccaro et al., 2004; Vedral et al., 1996; Skoneczny et al., 2008) which consider the input C_{in}. Thapliyal and Ranganathan (2011) presented an adder which optimizes the reduction of the computation in the ripple-carry process thanks to the use of a new gate called *TR*. This gate was defined in Thapliyal and Ranganathan (2009), but in Thapliyal and Ranganathan (2011) is optimized, using only 1 *CNOT* gate, 2 Controlled-*V* gates and 1 Controlled-V^+ gate. It has a quantum cost of 4 and a delay of $4\triangle$. The resulting adder has a quantum cost of $15N - 6$, a delay of $9N + 5\triangle$, and neither auxiliary inputs nor garbage outputs (Fig. 20). The ripple-carry adder of Nagamani et al. (2014) improves the quantum cost and delay of Thapliyal and Ranganathan (2011) ($12N$ and $10N\triangle$ respectively) at the cost of using $4N$ auxiliary inputs. The optical reversible ripple-carry adder with C_{in} proposed in Kotiyal (2016) is remarkable. It is not better than (Thapliyal and Ranganathan, 2011) in terms of the metrics of Mohammadi et al. (2009), but it improves it in terms of optical cost. Optical cost is one of the most important metric parameters in optical computing.

Moving away from the goal of reducing the quantum cost, Gidney presented an alternative gate to the Toffoli gate focused on reducing the cost of T gates (Gidney, 2018). This new gate has a higher quantum cost than the Toffoli gate if we consider the version proposed in Nielsen and Chuang (2011). However, it improves upon Toffoli gate implementations focused on fault-tolerance. As an example of the advantages of this gate (called *temporary logical-AND*), the author proposes an implementation of a fault-tolerant adder. This circuit has a quantum cost of

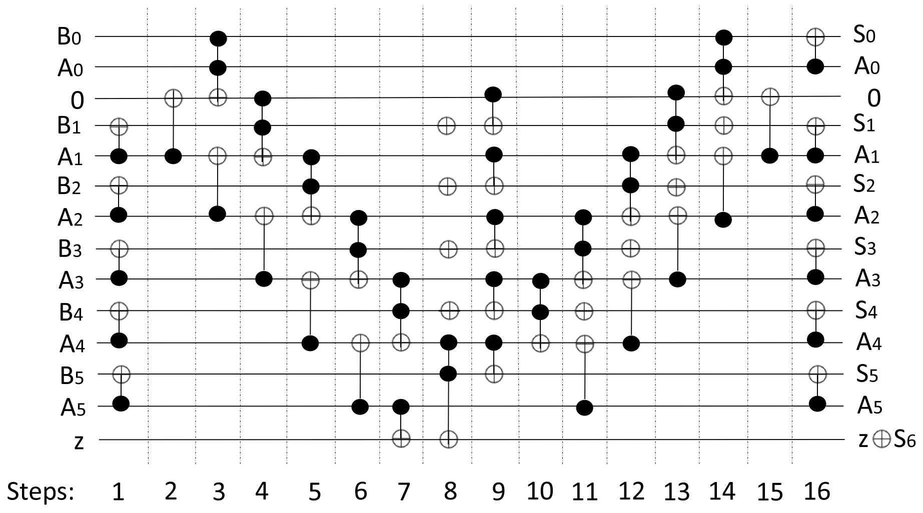


Fig. 18. Ripple-carry adder for $N = 6$ (assuming $C_{in} = 0$) proposed in Cuccaro et al. (2004).

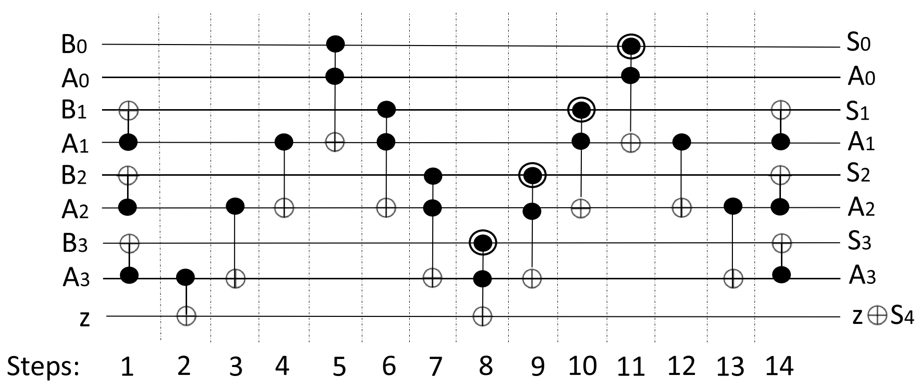


Fig. 19. Ripple-carry adder without carry in for $N = 4$ proposed in Thapliyal and Ranganathan (2013).

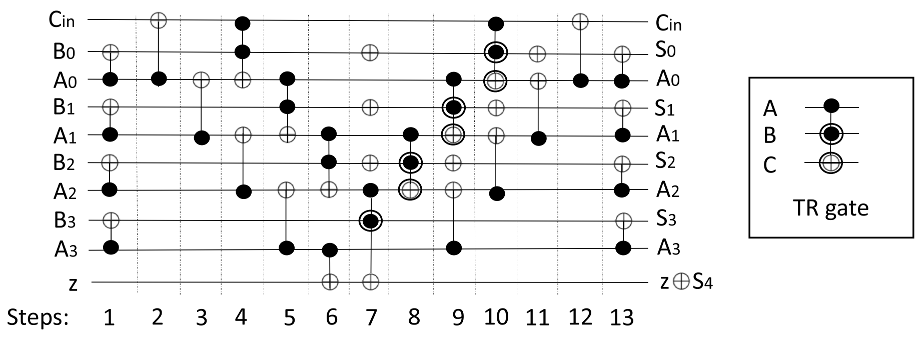


Fig. 20. Ripple-carry adder with carry in for $N = 4$ proposed in Thapliyal and Ranganathan (2011).

$18N - 2$, a delay of $15N - 5\triangle$, and requires N ancillary entries and 0 garbage outputs. Although these numbers are worse than previous circuits, that is due to their fault tolerance.

At the time of writing this article, the last adder in this category corresponds to the one published by Li et al. (Mohammadi et al., 2020). This adder combines Peres and *TR* gates to achieve the addition with a quantum cost of $13N - 10$, a delay of $10N - 4$, only 1 ancilla input and no garbage outputs. An example of this adder is shown in Fig. 21. The authors of this adder also describes its implementation in terms of T gates, obtaining an equivalent but error-oriented circuit. This second version has a quantum cost of $35N - 25$, a delay of $16N - 3$, and the same number of ancilla inputs and garbage outputs (1 and 0, respectively).

4.3.2. Carry-lookahead adder

This kind of adders employs two special signals to compute the carry out: generate signal (G) and propagate signal (P) (Harris and Harris, 2015):

- The carry out C_{out} of a pair of bits A_i and B_i is always 1 if both values are 1. This is called *generation* of a carry. Following this idea, G_i, the generate signal for the i-th pair, can be computed as $G_i = A_iB_i$.
- If a carry out C_{out} is produced when there is a carry in C_{in}, it is said that the carry is *propagated*. P_i, the propagate signal, can be computed as $P_i = A_i + B_i$.

Considering both signals, the carry out can be computed as:

$$C_i = A_iB_i + (A_i + B_i)C_{i-1} = G_i + P_iC_{i-1}$$

These adders are faster than the previous 1 s. However, they have a higher quantum cost since they need more operations to anticipate the computation of the *Gi* and *Pi* signals (Harris and Harris, 2015).

In 2004, Draper et al. proposed a logarithmic-depth reversible carry-lookahead adder which improves the delay of the previous linear -depth

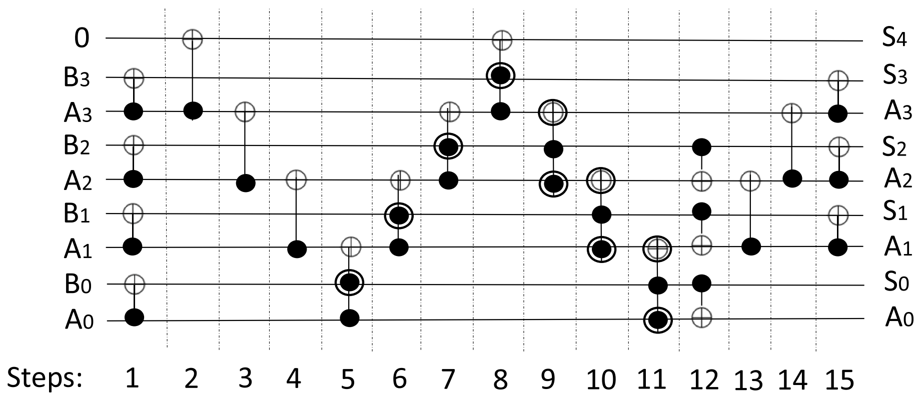


Fig. 21. Ripple-carry adder with carry in for $N = 4$ proposed in Mohammadi et al. (2020).

adders (Draper et al., 2004). It has a quantum cost of $28N - 15W(N) - 15\log(N) - 6$ (where $W(N)$ represents the number of ones in the binary expansion of N), a delay of $logN + logN/3 + 7$, $5N/4$ auxiliary inputs and no garbage outputs. Thapliyal et al. optimized the methodology to compute a carry-lookahead addition (without C_{in}), improving the quantum cost and delay of the previous adder (Thapliyal et al., 2013). It has a quantum cost of $26N - 15W(N) - 15\log(N-4)$ and a delay of $logN + logN/3 + 2$. The optimization is possible by computing G_i and P_i in parallel, and also replacing several *CNOT* and Toffoli gates by Peres gates. The circuit involves the use of several ancilla inputs, Zg_i and Zp_i, to compute G_i and P_i respectively. An example of this circuit is shown in Fig. 22. It works as follow s:

- Step 1: This step computes G_{i+1} and P_{i+1}, where $0 \leq i \leq N-1$. Zg_0 is transformed into $Zg_0 \oplus = A_0B_0$ with a Toffoli gate. For the case $i > 0$, $B_i \oplus = A_i$ and $Zg_i \oplus = A_iB_i$ using Peres gates.
- Step 2: This step computes G_{i+2} and P_{i+2}, where $0 \leq i \leq N-2$ for G and $2 \leq i \leq N-2$ for P. Using Toffoli gates, compute $Zp_i \oplus = B_iB_{i+1}$ for $i = 2$ to $N-2$ and $Zg_{i+1} \oplus = Zg_iB_{i+1}$ for $i = 0$ to $N-2$.
- Step 3: This step computes G_{i+3}, G_{i+4} and P_{i+4}, where $i = N/2$ and 0 for G and $i = N/2$ for P. For $i = N/2$ and $i = 0$, compute $Zp_{i+3} \oplus = Zg_{i+1}Zp_{i+1}$. When $i = N/2$, compute $Zp_{i+1} \oplus = Zp_iZp_{i+2}$.
- Step 4: This step computes G_{i+1} for $i = N$ and $i = N-2$. Compute $Zg_{i-1} \oplus = Zg_{i-3}Zp_{i-2}$ for $i = N/2$ and $Zg_{i-1} \oplus = Zg_{i-5}Zp_{i-3}$ for $i = N$. Also, this step uncomputes the values of Zp computed in step 3.
- Step 5: For $i = 2$ to $i = N-2$, transform Zg_i into $Zg_i \oplus = Zg_{i-1}B_i$, and Zp_{i+1} into $Zp_{i+1} \oplus = Zp_{i-1}Zp_{i+1}$ for $i = N/2$.
- Step 6: Uncompute Zp_i to avoid garbage outputs, and transform Zg_i into $Zg_i \oplus = B_{i+1}$ to compute S_{i+1}.
- Step 7: Compute S_0 and uncompute B_i.

Two reversible carry-lookahead adders were presented in Rahmati et al. (2017). They are shown (for the case $N = 4$) in Figs. 23 and 24. The first one has a quantum cost of $(2 \times N \times 4) + (N \times 1)$, better than the

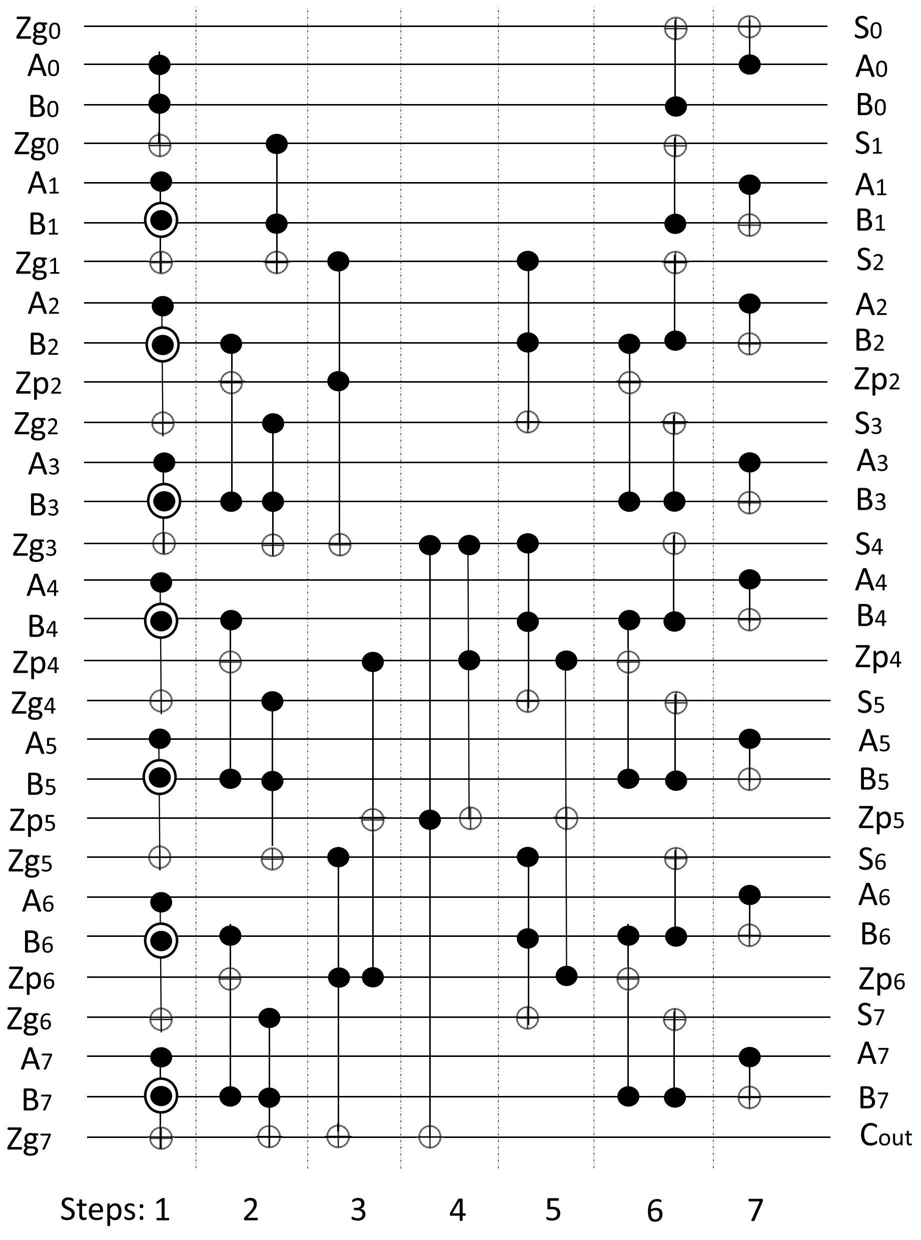


Fig. 22. Example of the carry-lookahead adder proposed in Thapliyal et al. (2013) for the case $N = 8$. It is built using Toffoli, *CNOT* and Peres gates. A_i and B_i are the numbers to be added, C_{out} are the carry out and S_i are the digits of the sum. Zg_i and Zp_i are auxiliary inputs used to compute G_i and P_i respectively.

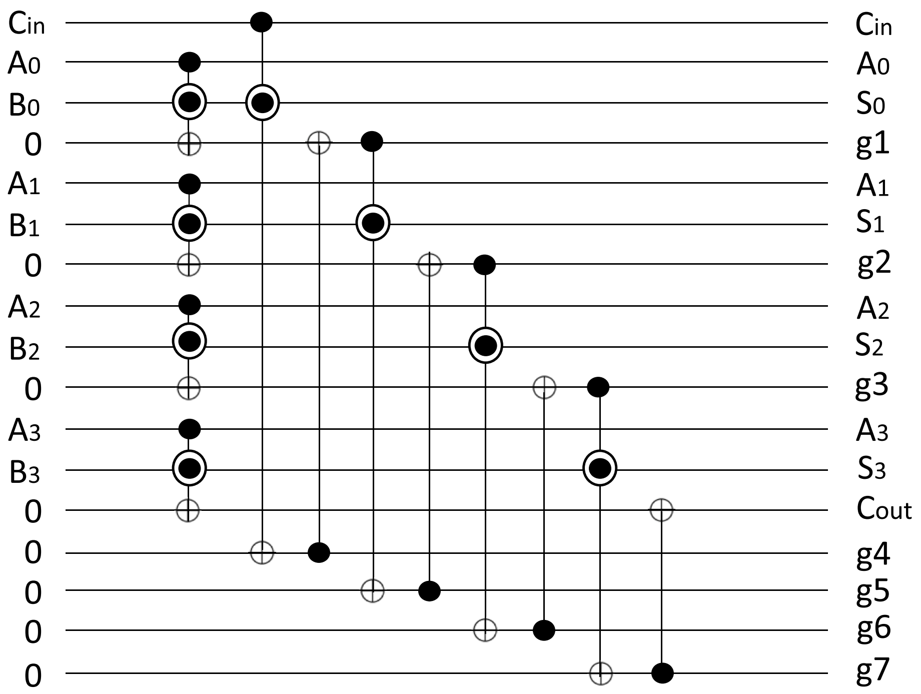


Fig. 23. Example of the first design of a carry-lookahead adder proposed in Rahmati et al. (2017) for the case $N = 4$. It is built using *CNOT* and Peres gates. A_i and B_i are the numbers to be added, C_{in} and C_{out} are the carry in and the carry out respectively, S_i are the digits of the sum, and g_i are garbage outputs.

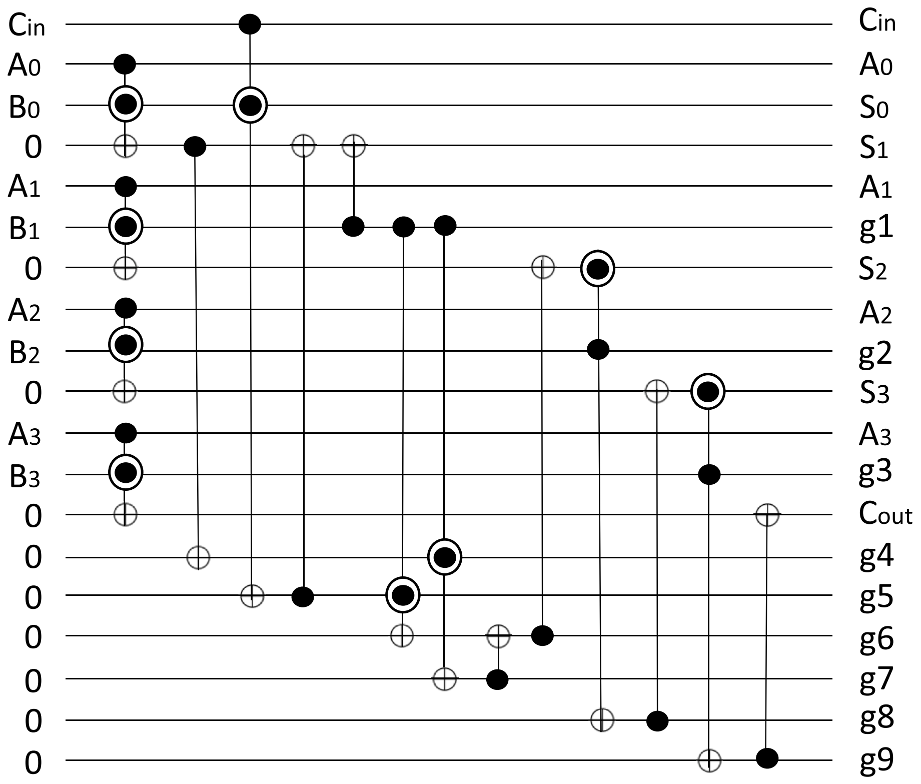


Fig. 24. Example of the second design of a carry-lookahead adder proposed in Rahmati et al. (2017) for the case $N = 4$. It is built using *CNOT* and Peres gates. A_i and B_i are the numbers to be added, C_{in} and C_{out} are the carry in and the carry out respectively, S_i are the digits of the sum, and g_i are garbage outputs.

presented in Thapliyal et al. (2013). However, this circuit presents $3 \times N$ garbage outputs, whereas the circuit of Thapliyal et al. (2013) has no garbage outputs. Following Bennett's garbage removal scheme (Bennett, 1973), it would be necessary to add $N + 1$ extra qubits to save S_i and C_{out}, four extra *CNOT* gates to copy S_i and C_{out} to those qubits, and to apply the reverse of the circuit. Therefore, uncomputing the garbage outputs would mean a final quantum cost of $2 \times ((2 \times N \times 4) + (N \times 1)) + N + 1$, which is higher than the quantum cost of Thapliyal et al. (2013). The second adder presented in Rahmati et al. (2017) also presents garbage outputs, so the same procedure can be applied to it. Its final quantum cost and delay do not improve that of Thapliyal et al. (2013).

A reversible adder presented in Lisa and Babu (2015) improves the quantum cost and delay of Thapliyal et al. (2013) using a novel technique for generating carry output. Nevertheless, it also has garbage outputs (Fig. 25). This adder was built using a new 4×4 gate called *RPA* (Reversible Partial Adder) gate, which has a quantum cost of 5 and delay $5\triangle$. It also uses several 4×4 Fredkin gates (the 3×3 Fredkin gate has a quantum cost of 5 and the same delay). The quantum cost is $11N$, but uncomputing the garbage outputs with (Bennett, 1973) would increase this number to $2 \times 11N + (N+1)$. For instance, for the case $N = 4$, this circuit would have a quantum cost of 93, whereas the adders of Rahmati et al. (2017) and Thapliyal et al. (2013) have 61 and 55 respectively.

Focusing on fault-tolerance, Thapliyal et al. presented in Thapliyal et al. (2020) four adders based on their own design of Thapliyal et al. (2013) but optimizing the number of T gates at the cost of increasing the rest of the metrics. To achieve this optimization they use the Gidney's *temporary logical-AND* gate. Through the four circuits they explore several combinations of temporary logical-AND and Toffoli gates to find the best possibilities in terms of T-count and number of ancilla inputs. At best, $2N - W(N) - \log(N) + 1$ ancilla inputs are required (as opposed to $5N/5$ of the original circuit). Their quantum cost and delay are much higher than the (Thapliyal et al., 2013) circuit. It is the cost to pay for fault tolerance.

5. Comparative analysis

In this section, the analyzed adders are compared. Since comparing adders of different types makes no sense, the comparison is carried out between adders of the same kind. Therefore, four comparison are presented: half adders, full adders, ripple-carry adders and carry-lookahead adders. Nevertheless, at the end of the section an overview of all the results is made to consider the comparison as a whole.

5.1. Half adders

Table 5 shows the quantum cost, delay, number of auxiliary inputs and number of auxiliary outputs of the most representative half adders. The final column indicates if the adder can be used as a subtractor. In terms of quantum cost, the Peres gate proposed in Hung et al. (2006) (Fig. 4) achieves the best value. The proposals of Sarma and Jain (2018) and Yamashita et al. (2008) have a quantum cost of 5, one more than (Hung et al., 2006), but (Sarma and Jain, 2018) can also act as a subtractor, so the extra value is justified in this case. Nielsen and Chuang (2011) has a quantum cost of 6, and Kaur and Dhaliwal (2012) has the highest quantum cost, 14. The quantum cost of Kaur and Dhaliwal (2012) is justified since this adder was the first reversible adder/-subtractor. None of the half adders can compute any operation in parallel, so their delay is equal to their quantum cost. In terms of auxiliary inputs, all of them have 1 input, except (Kaur and Dhaliwal, 2012) which has 3. Only (Kaur and Dhaliwal, 2012) presents garbage outputs.

Table 5
Comparative evaluation of half adders.

Adder	Quantum cost	Delay $\triangle$	Ancilla inputs	Garbage outputs	Adder/ subtractor
Kaur and Dhaliwal (2012)	14	14	3	3	Yes
Nielsen and Chuang (2011)	6	6	1	0	
Yamashita et al. (2008)	5	5	1	0	
Sarma and Jain (2018)	5	5	1	0	Yes
Hung et al. (2006)	4	4	1	0	

5.2. Full adders

Table 6 focus the comparison on the most relevant full adders. A new columns ha been added to this comparison in order to identify which adders are fault tolerant. The most optimized adder in terms of quantum cost, delay and garbage output is Thapliyal (2016) [a], with 6, $4\triangle$ and 0 respectively. The adder o f (Maslov et al., 2008) also presents the same quantum cost, delay and number of auxiliary inputs, but it has 1 garbage output. The only full adder which has no auxiliary inputs is Cuccaro et al. (2004), but it has a quantum cost and a delay higher than the average (14 and $14\triangle$ respectively). Bhagyalakshmi and Venkatesha (2010), Maslov et al. (2008), Singh and Rai (2016), Mitra and Chowdhury (2012), Zhou et al. (2014), Rangaraju et al. (2010), Saligram and Rakshith (2013) and Kumar et al. (2017) present garbage outputs, so their quantum cost and delay would be higher if they were uncomputed (Bennett, 1973). The best adder/subtractor is the proposed in Thapliyal (2016) [b] as it has no garbage outputs, and it has the best quantum cost and delay among the adder/subtractor s, 8 and $5\triangle$ respectively. It is followed by (Rangaraju et al., 2010) with a quantum cost of 10 and delay $10\triangle$, but with 3 garbage outputs. Therefore, the adder of Sarma and Jain (2018), which has a quantum cost of 15, would be a better option

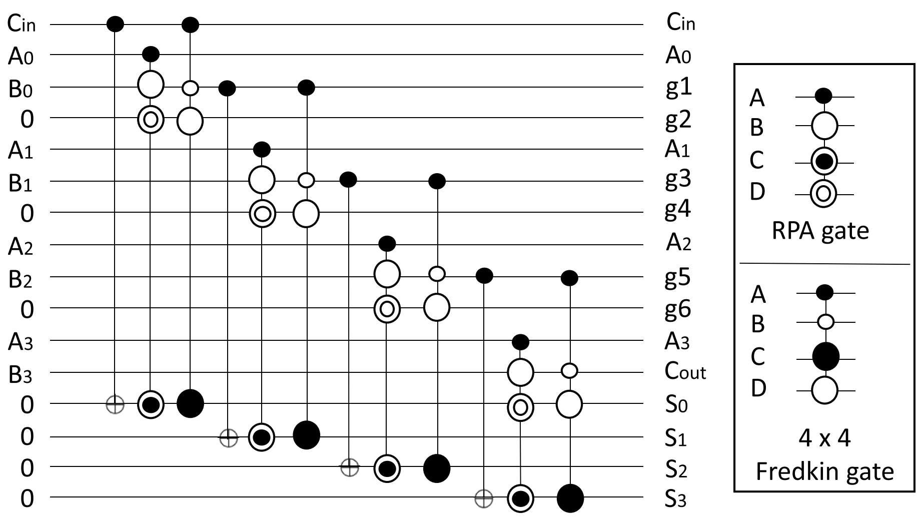


Fig. 25. Carry-lookahead adder proposed in Lisa and Babu (2015), for the case $N = 4$. It is built using 4×4 *RPA* and Fredkin gates. A_i and B_i are the numbers to be added, C_{in} and C_{out} are the carry in and the carry out respectively, S_i is the sum, and g_i are garbage outputs.

Table 6
Comparative evaluation of full adders.

Adder	Quantum cost	Delay $\triangle$	Ancilla inputs	Garbage outputs	Adder/subtractor	Fault tolerant
Saligram and Rakshith (2013)	18	18	5	6	Yes	
Yamashita et al. (2008)	15	15	3	0		
Sarma and Jain (2018)	15	10	1	0	Yes	
Cuccaro et al. (2004)	14	14	0	0		
Hung et al. (2006)	12	12	3	0		
Nagamani et al. (2014)	12	12	1	0		
Mitra and Chowdhury (2012)	11	11	2	3		Yes
Kumar et al. (2017)	11	11	4	4	Yes	
Wang et al. (2016)	10	8	1	0		
Rangaraju et al. (2010)	10	10	1	3	Yes	
Singh and Rai (2016)	8	8	1	1		
Bhagyalakshmi and Venkatesha (2010)	8	8	1	1		
Thapliyal (2016) [b]	8	5	2	0	Yes	
Zhou et al. (2014)	8	7	2	2		Yes
Maslov et al. (2008)	6	4	1	1		
Thapliyal (2016) [a]	6	4	1	0		

than (Rangaraju et al., 2010) as it has no garbage outputs. Finally, focusing on the fault tolerant adders, Zhou et al. (2014) and Mitra and Chowdhury (2012) are the most optimized options. Zhou et al. (2014) presents lower values of quantum cost and delay, and both have the same number of auxiliary inputs. Moreover, Zhou et al. (2014) has less garbage outputs.

5.3. Ripple-carry adders

The comparison between ripple-carry adders is shown in Table 7. A N-bits ripple-carry adder could be built chaining N full adders of Table 6. The better the selected full adder, the better the ripple-carry adder. However, even choosing (Thapliyal, 2016)[a] (the best full adder) results in a ripple-carry adder which is worse than, for instance, the improved ripple-carry adder presented in Thapliyal and Ranganathan (2013). For that reason, the resulting ripple-carry adders are not included in Table 7. In this table, the new column C_{in} indicates if the adder support s C_{in} or not.

In terms of auxiliary inputs, only Cuccaro et al. (2004) and Nagamani et al. (2014) have them. The adder of Cuccaro et al. (2004) has only 1, whereas the adder of Nagamani et al. (2014) employs $4N$. In terms of quantum cost, Nagamani et al. (2014) achieves the best value -even considering that it supports C_{in}- thanks to the use of the mentioned extra ancilla inputs. The non fault-tolerant adder of Mohammadi et al. (2020) improves the quantum cost of Nagamani et al. (2014), but only when $N < 10$. Thapliyal and Ranganathan (2013) is the third best adder in these terms. However, both Mohammadi et al. (2020) and Thapliyal and Ranganathan (2013) do not support C_{in}. In terms of delay, the circuit of Thapliyal and Ranganathan (2011) gets the best value, $9N + 5\triangle$, followed by the non fault-tolerant adder of Mohammadi et al. (2020), which has $10N - 4\triangle$. Finally, we can highlight that no circuit has garbage outputs.

The adder presented by Gidney (2018) is optimized for fault tolerance, and that is why it has higher values of quantum cost and delay. It even sacrifices multiple ancillary inputs to reduce the number of T gates. The fault-tolerant adder proposed in Mohammadi et al. (2020) does not improve the quantum cost nor the delay with respect to the previous one, but it represents a substantial improvement in terms of ancilla inputs.

Table 8
Comparative evaluations of carry-lookahead adders. $W(N)$ is the number of ones in the binary expansion of N. The quantum cost and delay of the circuits of Thapliyal et al. (2020) cannot be precisely indicated because certain transformations in the quantum state of the ancilla qubits need to be taken into account for use with temporary logical-AND gates. The study of how these transformations can influence the logarithmic propagation of the adders is not trivial.

Adder	Quantum cost	Delay $\triangle$	Ancilla inputs	Fault-tolerance
Thapliyal et al. (2020)[a]	$>40N$	$O(logN)$	$4N - 2W(N) - 2\log(N)$	Yes
Thapliyal et al. (2020)[b]	$>40N$	$O(logN)$	$2N - W(N) - \log(N) + 1$	Yes
Thapliyal et al. (2020)[c]	$>40N$	$O(logN)$	$4N - 2W(N) - 2\log(N)$	Yes
Thapliyal et al. (2020)[d]	$>40N$	$O(logN)$	$2N - W(N) - \log(N) + 1$	Yes
Draper et al. (2004)	$28N - 15W(N) - 15\log(N) - 6$	$logN + logN/3 + 7$	$5N/4$	
Thapliyal et al. (2013)	$26N - 15W(N) - 15\log(N - 4)$	$logN + logN/3 + 2$	$5N/4$	

Table 7
Comparative evaluation of ripple-carry adders.

Adder	Quantum cost	Delay $\triangle$	Ancilla inputs	Garbage outputs	C_{in}	Fault tolerant
Li et al. (2) (Mohammadi et al., 2020)	$35N - 25$	$16N - 3$	0	0		Yes
Takahashi and Kunihiro (2005)	$26N - 29$	$24N - 27$	0	0		
Gidney (2018)	$18N - 2$	$15N - 5$	N	0		Yes
Cuccaro et al. (2004)	$17N - 12$	$10N$	1	0		
Takahashi et al. (2010) (2)	$15N - 9$	$13N - 7$	0	0		
Thapliyal and Ranganathan (2011)	$15N - 6$	$9N + 5$	0	0	Yes	
Thapliyal and Ranganathan (2013) (2)	$13N - 8$	$11N - 4$	0	0		
Li et al. (1) (Mohammadi et al., 2020)	$13N - 10$	$10N - 4$	0	0		
Nagamani et al. (2014)	$12N$	$10N$	$4N$	0	Yes	

5.4. Carry-lookahead adders

Finally, the carry-lookahead adders are compared in Table 8. As it has been mentioned in the subsection of the carry-lookahead adders, there are several adders whose garbage outputs have not been uncomputed. They are the two adders of Rahmati et al. (2017) and Lisa and Babu (2015). It is not useful to compare such circuits with those which have uncomputed their garbage outputs since uncomputing these outputs following (Bennett, 1973) would increase the quantum cost, delay and auxiliary inputs (Orts et al., 2019). On the other hand, a 4 × 4 Fredkin gate is used in the case of Lisa and Babu (2015). The quantum cost and delay of this gate is not addressed, so it is not possible to determine the quantum cost and delay of this circuit with precision. Therefore, they have not been included in the table (but they have been analyzed in the subsection of the carry-lookahead adders for the sake of clarity). Considering the remaining non fault-tolerant adders, Draper et al. (2004) and Thapliyal et al. (2013), it can be concluded that Thapliyal et al. (2013) presents the best quantum cost and delay. Both of them have $5N/4$ auxiliary inputs and 0 garbage outputs. Considering now the four adders presented in Thapliyal et al. (2020), they do not improve any of the metrics from Mohammadi et al. (2009) to the previous adders. However, they are optimized in terms of the T gate, being the only adders focused on fault-tolerance in this category. In Table 8, the quantum cost and delay of these four adders are not shown accurately: this is because certain transformations in the quantum state of their ancilla qubits need to be taken into account for use with temporary logical-AND gates. The influence of these transformations in the quantum cost and delay of the adders is not trivial, and their effect is not included in the analysis done in citethapliyal2020tcount as it is focused on the optimization of T gates and necessary qubits.

5.5. General discussion

From the tables of the comparative evaluations, it can be concluded something that it is already known in classical circuits: ripple-carry adders have a lower cost, and carry-lookahead adders are the fastest. On the one hand, Mohammadi et al. (2020) and Thapliyal and Ranganathan (2013) for the case without C_{in} and Nagamani et al. (2014) and Thapliyal and Ranganathan (2011) for the case with C_{in} are the most optimized ripple-carry adders nowadays in terms of quantum cost, delay, auxiliary inputs and garbage outputs. On the other hand, Thapliyal et al. (2013) is the most optimized carry-lookahead adder. On the other hand, in the most recent works there is a growing interest in the optimization of circuits in terms of T gates. (Mohammadi et al., 2020; Gidney, 2018) (ripple-carry adders), and the four adders of Thapliyal et al. (2020) (carry-lookahead adders), are the best exponents in this new stage of optimization.

6. Conclusions

In this work, a revision on the state-of-the-art reversible adders has been carried out. First, appropriate metrics have been considered for the measurement and comparison of quantum circuits. Second, the adders have been classified in one of the four possible types: half adders, full adders, ripple-carry adders, and carry-lookahead adders, explaining their particular calculation methods and structures. Third, a complete analysis of each existing reversible adder has been done in terms of those metrics. Finally, a comparison between the analyzed adders has been done using the metrics and the category to determine which adders are the most beneficial in terms of quantum cost, delay, number of auxiliary inputs and/or number of garbage outputs, and taking into account the peculiarities of the category in question (if any) and fault tolerance.

The analysis has been carried out with two essential goals in mind: first, to collect and compare, using a set of standard metrics, all the reversible adders existing in the literature; second, to focus on the possibility of applying these adders in quantum circuits. To this extent, our emphasis has been on analyzing quantum costs and delays, as well as the presence of garbage outputs (which prevent useful interference from arising in quantum algorithms) and in presenting our findings in a clear and concise way, with tables that summarize the main properties of all the most important adders and that can be used as a quick reference by the interested researchers. In addition, we also provide figures that exemplify some of the most remarkable adders for some values on the number of inputs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

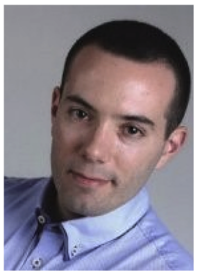
Acknowledgments

This work has been partially supported by the Spanish Ministry of Science throughout Project RTI2018-095993-B-I00, by the Regional Ministry of the Principado de Asturias under grant FC-GRUPIN-IDI/2018/000226 and by the European Regional Development Fund (ERDF). F. Orts is supported by an FPI Fellowship (attached to Project TIN2015-66680-C2-1-R) from the Spanish Ministry of Education.

References

Akbar, E.P.A., et al., 2011. Novel design of a fast reversible Wallace sign multiplier circuit in nanotechnology. Microelectron. J. 42 (8), 973–981.
Anamika, R. Bhardwaj, 2018. Reversible logic gates and its performances. In: 2018 2nd International Conference on Inventive Systems and Control. ICISC, pp. 226–231.
Arute, F., et al., 2019. Quantum supremacy using a programmable superconducting processor. Nature 574 (7779), 505–510.
Balaji, B., et al., 2018. Full adder/subtractor using reversible logic. Int. J. Pure Appl. Math. 120 (6), 437–446.
Barbosa, G.A., 2006. Quantum half-adder. Phys. Rev. A 73, 052321.
Batish, K., et al., 2018. Comparative analysis for performance evaluation of full adders using reversible logic gates. In: 2018 International Conference on Intelligent Circuits and Systems (ICICS). IEEE, pp. 126–132.
Bennett, C.H., 1973. Logical reversibility of computation. IBM J. Res. Dev. 17 (6), 525–532.
Bhagyalakshmi, H., Venkatesha, M., 2010. An improved design of a multiplier using reversible logic gates. Int. J. Eng. Sci. Technol. 2 (8), 3838–3845.
Bhuvana, B., VS, K.B., 2016. Design of reversible adders using a novel reversible bkg gate. In: 2016 Online International Conference on Green Engineering and Technologies (IC-GET). IEEE, pp. 1–6.
Bruce, J., et al., 2002. Efficient adder circuits based on a conservative reversible logic gate. In: Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI, IEEE, pp. 83–88, 2002.
Chatterjee, D., Roy, A., 2015. A transmon-based quantum half-adder scheme. Prog. Theor. Exp. Phys. (9).
Chaves, J., et al., 2018. Energy efficient QCA circuits design: simulating and analyzing partially reversible pipelines. J. Comput. Electron. 17 (1), 479–489.
Cohen, E., et al., 2016. All-optical design for inherently energy-conserving reversible gates and circuits. Nat. Commun. 7, 11424.
Combarro, E.F., et al., 2019a. A quantum algorithm for the commutativity of finite dimensional algebras. IEEE Access 7, 45554–45562.
Combarro, E.F., et al., 2019b. Quantum walks for the determination of commutativity of finite dimensional algebras. J. Comput. Appl. Math. 354, 496–506.
Cuccaro, S.A., Moulton, D.P., et al., 2004. A New Quantum Ripple-Carry Addition Circuit. arXiv:quant-ph/0410184.
Dastan, F., Haghparast, M., 2011. A novel nanometric fault tolerant reversible divider. Int. J. Phys. Sci. 6 (24), 5671–5681.
Diffie, W., Hellman, M., 1976. New directions in cryptography. IEEE Trans. Inf. Theor. 22 (6), 644–654.
Draper, T.G., et al., 2004. A Logarithmic-Depth Quantum Carry-Lookahead Adder. arXiv: quant-ph/0406142.
Dridi, G., et al., 2015. The mathematics of a quantum Hamiltonian computing half adder boolean logic gate. Nanotechnology 26 (34), 344003.
Eloie, L., et al., 2018. Quantum arithmetics via computation with minimized external control: the half-adder. Phys. Rev. A 97, 062321.
Fowler, A.G., et al., 2004. Implementation of Shor's algorithm on a linear nearest neighbour qubit array. Quant. Inf. Comput. 4 (4), 237–251.
Fredkin, E., Toffoli, T., 1982. Conservative logic. Int. J. Theor. Phys. 21 (34), 219–253.
Gidney, C., 2018. Halving the cost of quantum addition. Quantum 2 (74), 10–22331.
Grover, L.K., 1996. A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96. ACM, New York, NY, USA, pp. 212–219.

Groe, D., et al., 2008. Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares. In: 38th International Symposium on Multiple Valued Logic (ismvl 2008). IEEE, pp. 214–219.
Haghparast, M., Navi, K., 2008. Design of a novel fault tolerant reversible full adder for nanotechnology based systems. World Appl. Sci. J. 3 (1), 114–118.
Harris, S., Harris, D., 2015. Digital Design and Computer Architecture, arm edition. Morgan Kaufmann.
He, X., et al., 2019. The mapping and optimization method of quantum circuits for Clifford+T gate. J. Appl. Math. Phys. 7 (11), 2796–2810.
Hung, W.N., et al., 2006. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 25 (9), 1652–1663.
Islam, M., et al., 2009a. Efficient approaches for designing fault tolerant reversible carry look-ahead and carry-skip adders. MASAUM J. Basic Appl. Sci. 1 (3), 354–360.
Islam, M.S., et al., 2009b. Fault tolerant reversible logic synthesis: carry look-ahead and carry-skip adders. In: 2009 International Conference on Advances in Computational Tools for Engineering Applications. IEEE, pp. 396–401.
Islam, M.S., et al., 2009c. Synthesis of fault tolerant reversible logic circuits. In: 2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis. IEEE, pp. 1–4.
Kaur, P., Dhaliwal, B.S., 2012. Design of fault tolerant full adder/subtractor using reversible gates. In: 2012 International Conference on Computer Communication and Informatics. IEEE, pp. 1–5.
Khlopotine, A.B., et al., 2002. Reversible logic synthesis by iterative compositions. In: Iwls, pp. 261–266.
Kotiyal, S., 2016. Design Exploration and Application of Reversible Circuits in Emerging Technologies. Ph.D. thesis. Computer science and engineering department. University of South Florida.
Krishnaveni, D., et al., 2012. Design of an efficient reversible 8x8 Wallace tree multiplier. World Appl. Sci. J. 20 (8), 1159–1165.
Kumar, P.K., et al., 2017. Optimal design of reversible parity preserving new full adder/ full subtractor. In: 2017 11th International Conference on Intelligent Systems and Control (ISCO). IEEE, pp. 368–373.
Landauer, R., 1961. Irreversibility and heat generation in the computing process. IBM J. Res. Dev. 5 (3), 183–191.
Lanyon, B.P., et al., 2009. Simplifying quantum logic using higher-dimensional hilbert spaces. Nat. Phys. 5 (2), 134.
Lecerf, Y., 1963. Machines de Turing rversibles. Comptes Rendus Hebd. Sances Acad. Sci. 257, 2597–2600.
Lemr, K., et al., 2015. Experimental implementation of optimal linear-optical controlled-unitary gates. Phys. Rev. Lett. 114 (15), 153602.
Linke, N.M., et al., 2017. Experimental comparison of two quantum computing architectures. Proc. Natl. Acad. Sci. 114 (13), 3305–3310.
Lisa, N.J., Babu, H.M.H., 2015. Design of a compact reversible carry look-ahead adder using dynamic programming. In: 2015 28th International Conference on VLSI Design. IEEE, pp. 238–243.
Maslov, D., Negrevergne, C., et al., 2008. Quantum circuit simplification and level compaction. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 27 (3), 436–444.
Meter, R.V., et al., 2008. Arithmetic on a distributed-memory quantum multicomputer. ACM J. Emerg. Technol. Comput. Syst. (JETC) 3 (4), 2.
Michielse, K., et al., 2017. Benchmarking gate-based quantum computers. Comput. Phys. Commun. 220, 44–55.
Miller, D.M., et al., 2014. Mapping NCV circuits to optimized Clifford+T circuits. In: International Conference on Reversible Computation. Springer, pp. 163–175.
Mitra, S.K., Chowdhury, A.R., 2012. Minimum cost fault tolerant adder circuits in reversible logic synthesis. In: 2012 25th International Conference on VLSI Design. IEEE, pp. 334–339.
Mohammadi, M., et al., 2009. On figures of merit in reversible and quantum logic designs. Quant. Inf. Process. 8 (4), 297–318.
Mohammadi, M., et al., 2020. Efficient quantum arithmetic operation circuits for quantum image processing. Sci. China Phys. Mech. Astron. 63, 1–13.
Nagamani, A., et al., 2014. Design of optimized reversible binary adder/subtractor and BCD adder. In: 2014 International Conference on Contemporary Computing and Informatics (IC3I). IEEE, pp. 774–779.
Neill, C., et al., 2018. A blueprint for demonstrating quantum supremacy with superconducting qubits. Science 360 (6385), 195–199.
Nielsen, M.A., Chuang, I.L., 2011. Quantum Computation and Quantum Information, 10th anniversary edition.
Orts, F., et al., 2019. An optimized quantum circuit for converting from signmagnitude to two's complement. Quant. Inf. Process. 18 (11), 332.
Parhami, B., 2006. Fault-tolerant reversible circuits. In: 2006 Fortieth Asilomar Conference on Signals, Systems and Computers. IEEE, pp. 1726–1729.
Pavlidis, A., Gizopoulos, D., 2014. Fast quantum modular exponentiation architecture for Shor's factorization algorithm. Quant. Inf. Comput. 14 (7), 649–682.
Peres, A., 1985. Reversible logic and quantum computers. Phys. Rev. A 32 (6), 3266.
Rahmati, M., et al., 2017. Novel designs of a carry/borrow look-ahead adder/subtractor using reversible gates. J. Comput. Electron. 16 (3), 856–866.
Rangaraju, H., et al., 2010. Low power reversible parallel binary adder/subtractor. Int. J. VLSI Des. Commun. Syst. 1 (3), 23–34.
Rivest, R.L., et al., 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21 (2), 120–126.
Rohini, H., Rajashekar, S., 2016. Design of reversible logic based basic combinational circuits. Commun. Appl. Electron. 5 (9), 38–43.
Sahu, L., Kumar, U., Singh, L., 2019. Implementation of improved energy-efficient FIR filter using reversible logic. In: Data Science and Big Data Analytics. Lecture Notes on Data Engineering and Communications Technologies, vol. 16. Springer, Singapore, pp. 229–238.
Saligram, R., Rakshith, T., 2013. Design of low logical cost adders using novel parity conserving toffoli gate. In: 2013 International Conference on Emerging Trends in Communication, Control, Signal Processing and Computing Applications (C2SPCA). IEEE, pp. 1–6.
Sarma, R., Jain, R., 2018. Quantum gate implementation of a novel reversible half adder and subtractor circuit. In: 2018 International Conference on Intelligent Circuits and Systems (ICICS). IEEE, pp. 72–76.
Shi, Y., 2003. Both Toffoli and controlled-not need little help to do universal quantum computing. Quant. Inf. Comput. 3 (1), 84–92.
Shor, P.W., 1994. Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, SFCS '94. IEEE Computer Society, Washington, DC, USA, pp. 124–134.
Singh, V.P., Rai, M., 2016. Verilog design of full adder based on reversible gates. In: 2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall). IEEE, pp. 1–5.
Skoneczny, M., et al., 2008. Reversible fourier transform chip. In: 2008 15th International Conference on Mixed Design of Integrated Circuits and Systems. IEEE, pp. 281–286.
Srivastava, S., et al., 2017. Quantum half-adder boolean logic gate with a nano-graphene molecule and graphene nano-electrodes. Chem. Phys. Lett. 667, 301–306.
Steiger, D.S., et al., 2018. ProjectQ: an open source software framework for quantum computing. Quantum 2 (49).
Takahashi, Y., Kunihiro, N., 2005. A linear-size quantum circuit for addition with no ancillary qubits. Quant. Inf. Comput. 5 (6), 440–448.
Takahashi, Y., Kunihiro, N., 2008. A fast quantum circuit for addition with few qubits. Quant. Inf. Comput. 8 (6), 636–649.
Takahashi, Y., et al., 2010. Quantum addition circuits and unbounded fan-out. Quant. Inf. Comput. 10 (9), 872–890.
Thapliyal, H., 2016. Mapping of subtractor and adder-subtractor circuits on reversible quantum gates. In: Transactions on Computational Science XXVII. Springer, pp. 10–34.
Thapliyal, H., Ranganathan, N., 2009. Design of efficient reversible binary subtractors based on a new reversible gate. In: 2009 IEEE Computer Society Annual Symposium on VLSI. IEEE, pp. 229–234.
Thapliyal, H., Ranganathan, N., 2010. Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs. ACM J. Emerg. Technol. Comput. Syst. (JETC) 6 (4), 14.
Thapliyal, H., Ranganathan, N., 2011. A new reversible design of bcd adder. In: 2011 Design, Automation & Test in Europe. IEEE, pp. 1–4.
Thapliyal, H., Ranganathan, N., 2013. Design of efficient reversible logic-based binary and bcd adder circuits. ACM J. Emerg. Technol. Comput. Syst. (JETC) 9 (3), 17.
Thapliyal, H., et al., 2020. T-count and Qubit Optimized Quantum Circuit Designs of Carry Lookahead Adder. arXiv:2004.01826.
Thapliyal, H., et al., 2013. Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder. In: Transactions on Computational Science XVII. Springer, pp. 73–97.
Toffoli, T., 1980. Reversible computing. In: Proceedings of the 7th Colloquium on Automata, Languages and Programming. Springer-Verlag, Berlin, Heidelberg, pp. 632–644.
Trisetyarso, A., Van Meter, R., 2010. Circuit design for a measurement-based quantum carry-lookahead adder. Int. J. Quant. Inf. 8, 843–867, 05.
Valinataj, M., et al., 2016. Novel low-cost and fault-tolerant reversible logic adders. Comput. Electr. Eng. 53, 56–72.
Vartiainen, J.J., et al., 2004. Implementing Shor's algorithm on Josephson charge qubits. Phys. Rev. A 70, 012319.
Vedral, V., et al., 1996. Quantum networks for elementary arithmetic operations. Phys. Rev. A 54 (1), 147.
Venegas-Andraca, S.E., 2012. Quantum walks: a comprehensive review. Quant. Inf. Process. 11 (5), 1015–1106.
Wang, F., et al., 2016. Improved quantum ripple-carry addition circuit. Sci. China Inf. Sci. 59 (4), 042406.
Wu, C., Cain, C., 2014. A non-qubit quantum adder as one-dimensional cellular automaton. Phys. E Low-dimens. Syst. Nanostruct. 59, 243–247.
Yamashita, S., et al., 2008. DDMF: an efficient decision diagram structure for design verification of quantum circuits under a practical restriction. IEICE Trans. Fund. Electron. Commun. Comput. Sci. 91 (12), 3793–3802.
Zhang, D., et al., 2018. Novel optimized link state routing protocol based on quantum genetic strategy for mobile learning. J. Netw. Comput. Appl. 122, 37–49.
Zhou, R.G., et al., 2014. Novel designs for fault tolerant reversible binary coded decimal adders. Int. J. Electron. 101 (10), 1336–1356.
Zhou, L., et al., 2018. Quantum technique for access control in cloud computing ii: encryption and key distribution. J. Netw. Comput. Appl. 103, 178–184.



F. Orts is a predoctoral researcher at the Informatics Department at the University of Almería, Spain. He studied the Master in Computer Engineering at the University of Almería. He is currently doing his PhD thanks to the Spanish FPI program. His publications and more information about him can be found in %http://hpca.ual.es/forts/. His research interests are Multi-Dimensional Scaling, Quantum Computing and High Performance Computing.

G. Ortega received the Ph.D. degree from the University of Almería (Spain) in 2014. From 2009, she has been working as a member of the TIC-146 Supercomputing-Algorithms research group. She is a PhD Assistant Professor at the Informatics Department at the University of Almería. Her current research work is focused on High Performance Computing and Optimization. Some of her research interest include the study of strategies for load balancing the workload on heterogeneous systems, the parallelization of optimization problems and image processing.

E.F. Combarro received the B.S. degree in mathematics (1997), the M.S. degree in computer science (2002), and the PhD In mathematics (2001) from the University of Oviedo (Spain), where he currently is an Associate Professor at the Computer Science Department. He has authored more than 40 research papers in topics such as computability theory, the theory of fuzzy measures, the computational classification of semifields and text categorization. His current research focus mainly in quantum computing and its applications to algebraic and optimization problems.

E.M. Garzón received her B.Sc. degree in Physics in 1985 from the University of Granada (Spain) and her PhD degree in Computer Engineering in 2000 from the University of Almería (Spain). From 1989 to 1993 she was an Assistant Professor at University of Granada. She is a Full Professor of the Department of Informatics at the University of Almería. Currently, she is head of the Supercomputing-Algorithms research group. Her research interest lies in the field of High-Performance Computing for irregular problems related to Matrix Computation, Image Processing and Global Optimization on heterogeneous HPC platforms and Quantum Computers.

2.2 Microrheology

This section consists on the following papers:

F. Orts, G. Ortega, E.M. Garzón and A.M. Puertas. Finite size effects in active microrheology in colloids. *Computer Physics Communications*, 236, 8-14, 2019. JCR (2019) = 3.627. Subject categories = Physics, Mathematical: 3/55 (**Q1**); Computer Science, Interdisciplinary Applications: 31/109 (Q2).

F. Orts, G. Ortega, E.M. Garzón, M. Fuchs and A.M. Puertas. Dynamics and friction of a large colloidal particle in a bath of hard spheres: Langevin dynamics simulations and hydrodynamic description. *Physical Review E*, 101(052607), 1-10, 2020. JCR (2019) = 2.296. Subject categories = Physics, Mathematical: 9/55 (**Q1**); Physics, Mathematical: 13/34 (Q2).

F. Orts, G. Ortega, A.M. Puertas, I. García and E.M. Garzón. On solving the unrelated parallel machine scheduling problem: active microrheology as a case study. *Journal of Supercomputing*, 76, 8494-8509, 2020. JCR (2020) = 2.474. Subject categories = Computer Science, Theory & Methods: 33/110 (**Q2**); Computer Science, Hardware & Architecture: 26/53 (Q2); Engineering, Electrical & Electronic: 139/273 (Q3).

E.F. Combarro, I.F. Rúa, **F. Orts**, G. Ortega, A.M. Puertas, and E.M. Garzón. Quantum Algorithms to compute the neighbour list of N-body Simulations. *Submitted and under review*, 2021.

2.2.1 Finite size effects in active microrheology in colloids

Title	Finite size effects in active microrheology in colloids		
Authors	*F. Orts, G. Ortega, E.M. Garzón and A.M. Puertas*		
Journal	Computer Physics Communications		
Year	2019		
Volume	236		
Pages	8-14		
DOI	https://doi.org/10.1016/j.cpc.2018.10.003		
IF (JCR 2019)	3.627		
Categories	Physics, Mathematical:	3/55	**(Q1)**
	Computer Science, Interdisciplinary Applications:	31/109	(Q2)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, has helped with the computation associated to this paper.

Computer Physics Communications 236 (2019) 8–14

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Finite size effects in active microrheology in colloids

F. Orts [a], G. Ortega [b], E.M. Garzón [a], A.M. Puertas [c,*]

[a] Supercomputation-Algorithms Group, Department of Informatics, University of Almería, ceiA3, Ctra. Sacramento s/n, 04120, Almería, Spain
[b] Computer Architecture Department, Universidad de Málaga, E.T.S.I. Informática Campus Teatinos, 29071, Málaga, Spain
[c] Group of Complex Fluids Physics, Department of Applied Physics, University of Almería, Ctra. Sacramento s/n, 04120, Almería, Spain

A R T I C L E I N F O

Article history:
Received 5 April 2018
Received in revised form 1 October 2018
Accepted 6 October 2018
Available online 25 October 2018

Keywords:
Microrheology
Finite size effects
Scheduling in CPU–GPU clusters
Simulations of colloids

A B S T R A C T

Active microrheology has emerged in recent years as a new technique to probe microscopically the mechanical properties of materials, particularly, viscoelastic ones. In this technique, a colloidal tracer is pulled through the material, and its dynamics is monitored. The interpretation of results usually relies on the Stokes–Einstein approximation, which is valid for a continuous medium in equilibrium. In this work, we have studied with simulations a suspension of quasi-hard colloidal spheres, where a large tracer is pulled by a constant force. The Navier–Stokes equation for a continuous bath predicts important finite size effects, decaying as the inverse box size, which require simulations of different systems to extract the microviscosity of a bulk system. A strategy to optimize the scheduling of the simulation tasks on a multi GPU–CPU cluster based on the adaptation of a genetic algorithm is presented here, and used to study the effect of different conditions on the friction experienced by the tracer (adding the tracer volume to the total system volume, fixing the center of mass of the system, varying the fluid friction coefficient and tracer size). It is observed that the theoretical prediction is not followed, but deviations are observed for large systems in all cases. These are attributed to the finite size of the bath particles, and the intrinsic dynamics of colloidal systems, as shown by the analysis of the velocity profile in the bath.

1. Introduction

Soft matter is characterized by the interplay of very different length and time scales. Physically, this is achieved, e.g., in suspensions of macromolecules or colloids, where the solvent degrees of freedom and macromolecule diffusion extend over many different scales [1,2]. This poses a major problem in resolving all of them, both computationally and experimentally, which is typically tackled integrating out the fast degrees of freedom, or using effective models [3]. The canonical example is probably Brownian hard spheres, where internal degrees of freedom are absent and the solvent dynamics is integrated in the so-called Brownian motion; still, the separation between short time and long time diffusion of the colloidal particles provokes viscoelastic behavior when the glass transition is approached.

In order to probe the complex mechanical behavior of soft matter, several techniques have been developed [4]. In addition to the direct measurement using bulk rheology, where macroscopic stresses are applied, microrheology has emerged over the last decades as a new methodology, both in the passive or active modes [5–8]. Here, the dynamics of a tracer (typically of colloidal size) is studied, with an external force driving it (active microrheology), or without any force (passive microrheology). This approach, initially thought for expensive or difficult to obtain samples [9,10], but still in development [11,12], requires a deep understanding of the interplay of the dynamics of bath and tracer, in particular due to the non-affine strain field provoked by the latter [13,14]. It is therefore compulsory to test the theory models to be used in the interpretation of the results.

Computer simulations have emerged as an ideal tool to test these models, using simple systems, whose bulk properties are well-known [7]. However, because simulations consider always a finite number of particles, finite-size effects can appear. This is even more plausible, since the hydrodynamic correlations, expected from models based on a continuum description of the bath [15], have a very long range. This makes the simulation work a formidable task because a large number of particles have to be considered to obtain the trajectory of a single tracer; even more, a large number of trajectories are needed to average out the thermal noise and initial conditions; and finally different system sizes have to be considered [16,17].

Previous simulation works in this line have focused on passive microrheology, i.e. the dynamics of a large unperturbed tracer, in a bath of hard spheres undergoing Newtonian dynamics [16–20]. In this way, those works have simulated the motion of a nanocolloidal particle, resulting in Brownian motion, and the fluctuation–dissipation theorem. The results show good qualitative agreement with the predictions from continuum theory for the bath, and when the tracer is larger than around five times the bath particles

* Corresponding author.
E-mail address: apuertas@ual.es (A.M. Puertas).

https://doi.org/10.1016/j.cpc.2018.10.003
0010-4655/© 2018 Elsevier B.V. All rights reserved.

the agreement is quantitative. Active microrheology in a bath of hard spheres, on the other hand, has been studied with tracers of size comparable to the bath particles, with Brownian, Langevin, or Stokesian microscopic dynamics [7,21–27]. Both the simulation and theoretical models show that the effective friction coefficient experienced by the tracer has a plateau for small pulling forces, where the properties of the bath are probed — linear response and generalized Stokes–Einstein relations are expected to be applicable here. Upon increasing the driving force, the effective friction enters a so-called force-thinning regime, where the coefficient decreases, until a plateau is eventually reached for strong pullings [28].

We present here simulations of active microrheology in a system of quasi-hard Brownian spheres with large tracers. These, however, are affected by strong finite size effects, requiring simulations of different sizes. Given the large number of independent simulations of different lengths, a multi GPU–CPU cluster was used, which supplies processors with several CPU-cores and GPUs. A sequential code to compute simulations on CPU-cores, and a parallelized one for computing on GPUs, prepared previously [29], have been used. To reduce the runtime for the set of simulations it is very important to distribute the different simulations among CPU-cores and GPUs in a balanced way, i.e. with a minimal idle processors time. This is a challenge due to the heterogeneity of simulations and computational power of CPU-cores and GPUs. A genetic algorithm has been implemented to obtain a near optimal balance in the distribution of simulations on the clusters, which is the result of this work (freely available at https://github.com/2forts/GENS). We focus on a system with a bath volume fraction of 50%, and a tracer three times larger than the bath particles. The scheme presented here optimizes the whole set of simulations required to analyze the friction coefficient. The acceleration of the single trajectories, due to the GPU parallelization, has allowed us to simulate large systems and study the effects of different parameters of the simulations of active microrheology in colloidal hard spheres, aiming to identify the optimal simulation conditions to test the theoretical model. For large systems, the simulation data deviate from the theoretical prediction, and the velocity field in the bath oscillates in phase with the density. More interestingly, we show that the velocity in the bath decays faster than predicted, and becomes negligible for distances similar to the simulation box size where the deviations appear.

The manuscript is organized as follows: In Section 2 the physical system is described, and the model used for its analysis introduced. Our scheme for the distribution of the tasks among the computational resources is described in Section 3. Section 4 is devoted to the presentation of the results, in particular analyzing the effects of the consideration of the volume of the tracer, the size of the tracer, or fixing the center of mass of the systems. In the final part of this section, we study the velocity profile induced in the bath by the moving tracer, and compare it with the theoretical predictions from the Navier–Stokes equation. Finally, the conclusions are presented in Section 5.

2. System details

Microrheology in a colloidal system is simulated considering N polydisperse Brownian particles containing the tracer (labeled with $j = 1$) in a cubic box with periodic boundary conditions. Microscopic Brownian dynamics is modeled with the Langevin equation of motion, which for particle j reads [30]:

$$m_j \frac{d^2 \mathbf{r}_j}{dt^2} = \sum_{i \neq j} \mathbf{F}_{ij} - \gamma_j \frac{d\,\mathbf{r}_j}{dt} + \mathbf{f}_j(t) + \mathbf{F}_{ext}\delta_{j1} \tag{1}$$

where m_j is the particle mass, and the terms in the right hand side correspond to the interaction forces between particles i and j, the friction with solvent, Brownian force and the external force, which acts only on the tracer (as shown by the Kronecker-delta symbol, δ_{j1}). The friction force is proportional to the particle velocity, and the proportionality constant is given by $\gamma_i = \gamma_0 a_i$, where a_i is the particle radius. This expression mimics the Stokes formula for spheres at low Reynolds numbers, $\gamma_i = 6\pi\eta a_i$, where η is the solvent viscosity. The Brownian force, $\mathbf{f}(t)$, is random, but its intensity is linked to the friction force, as given by the fluctuation–dissipation theorem, $\langle \mathbf{f}_j(t)\cdot\mathbf{f}_j(t')\rangle = 6k_BT\gamma_j\delta(t-t')$, where k_BT is the thermal energy and $\delta(x)$ is the Dirac-delta symbol [30].

The direct interaction between particles i and j is derived from the central inverse-power potential:

$$V(\mathbf{r}) = k_BT\left(\frac{r}{a_{ij}}\right)^{-36} \tag{2}$$

with $r = |\mathbf{r}|$ and a_{ij} the center to center distance between the particles. It has been shown previously that with this potential the particles behave effectively as hard spheres [31]. To avoid crystallization at high density, size polydispersity is introduced in the bath. Sizes for the bath particles are selected from a flat distribution of width $2\delta = 0.2a$, with a the mean radius of the bath particles. All particles, including the tracer, have the same mass: $m_j = m$.

In the simulations, the system is equilibrated with the tracer for a long time without the external force. At $t = 0$, the external force is switched on, and the tracer trajectory is recorded. The long-time steady tracer velocity, $\langle v\rangle$, is calculated as the slope of the tracer displacement vs. time, and averaged over many independent trajectories. This allows the calculation of the effective friction coefficient using the steady-state relationship $F_{\text{ext}} = \gamma_{\text{eff}}\langle v\rangle$. Previous simulations (with tracers of the same size as the bath particles) have shown that γ_{eff} develops a plateau for small forces, indicating a linear regime (Newtonian behavior) [7,21,22]. We intend to focus here in this linear regime for small forces, using tracers larger than the bath particles.

However, the use of periodic boundary conditions in the three dimensions implies that the actual simulated system is a cubic array of tracers pulled in an infinite bath. Because the lattice spacing in this array of tracers is given by the size of the simulation box, this is a finite size effect. In order to analyze, and eventually correct it, a continuum model based on solving the Navier–Stokes equation for an infinite array of tracers in a Newtonian viscous fluid is used [15]. Hasimoto [15] showed that the friction coefficient, γ_{eff}, experienced by a cubic array of tracers is related to the lattice spacing, L, as:

$$\frac{1}{\gamma_{\text{eff}}} = \frac{1}{\gamma_\infty}\left(1 - \frac{c}{L}\right) \tag{3}$$

where γ_∞ is the friction coefficient measured in an infinite system, and c is a constant that depends on the array structure (simple cubic, BCC, FCC, ...). In our case, due to the periodic boundary conditions, the simple cubic array applies, yielding $c = 2.8373\,a_t$ [15] with a_t the tracer size.

Based on this result, the full analysis of microrheology therefore requires simulations of systems with different sizes, to extrapolate the friction coefficient γ_∞ using Eq. (3), for a single value of the force, volume fraction, or tracer size. Because the simulation time for a system of N particles evolves as $\sim N \ln N$, it is important to state the validity of this extrapolation, which is the main aim of the present work.

Previous simulations in passive microrheology, i.e. without external force acting on the tracer, have shown that Eq. (3) describes the dependence of the diffusion coefficient ($D_{\text{eff}} = k_BT/\gamma_{\text{eff}}$) on the system size [16,17]. Evenmore, the value of γ_∞, extracted from the fitting corresponds to the Stokes value, with slip boundary conditions and the viscosity calculated from Green–Kubo integration of the stress autocorrelation function [16]. We check here if those

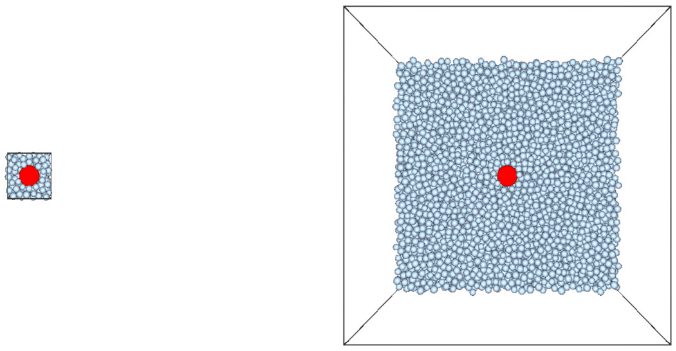


Fig. 1. Snapshots of the systems with $N = 216$ and 32768 particles (left and right panels, respectively), with the same scale. The tracer, with $a_t/a = 3$, is marked in red, and the particles in front of it have been removed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

conclusions are valid for a finite force, helping in understanding the generalized Stokes–Einstein relation. The application of an external force implies a continuous input of energy in the system, needing an energy sink, where this energy is dissipated, which is an important difference with respect to previous works in passive microrheology. In our case, energy is dissipated in the friction with the solvent, given by γ_0.

In order to test the theoretical model, we have run simulations of systems with $N = 216, 512, 1000, 2197, 4096, 8000, 15625$ and 32768 particles. Fig. 1 presents snapshots of the extreme sizes, with a tracer three times larger than the bath particles, $a_t = 3\,a$. In our simulations, lengths are measured in units of the mean bath particle radius, a, energy in units of the thermal energy $k_\mathrm{B}T$, and mass in units of the particle mass, m. For the friction coefficient with the solvent, we take $\gamma_0 = 5\sqrt{mk_\mathrm{B}T}/a$, which gives a mean single particle diffusion coefficient of $D_0 = k_\mathrm{B}T/\gamma_0 = 0.2\,a\sqrt{k_\mathrm{B}T/m}$ for the bath particles. The volume fraction of the bath is $\phi = 0.50$. The external force is applied in the x-axis. The equations of motion are integrated using the Heun algorithm [32], with a time step of $\delta t = 0.0005\,a\sqrt{m/k_\mathrm{B}T}$. In this algorithm, the friction force is integrated analytically in the time interval δt.

3. Computational implementation

From a computational point of view, the problem requires a large set of simulations of tracer trajectories in systems of different sizes (N); therefore, the use of high performance computing is mandatory.

In the model it is possible to identify two parallelism levels. Level 1 allows us to accelerate the computation of a single tracer trajectory; and level 2 is related to the computation of several trajectories. To compute the function $\gamma_0/\gamma_\mathrm{eff}(a/L)$ it is necessary to analyze the tracer dynamic for different sizes of the bath. Therefore, the second level of parallelism can be exploited executing simulations with different number of particles in parallel, needed for the extrapolation leading to γ_∞.

We have accelerated the computation of a single tracer trajectory (level 1) by means of GPU computing using CUDA interface [29,33]. Our attention has been focused on the acceleration of the routines which evaluate the tracer dynamics; mainly, the calculation of interaction forces and integration of the equations of motion [34]. Every simulation of the tracer dynamic includes a massive parallelism since the same computation has to be completed for all particles in the bath. This parallelism is harnessed by the simulations computed on GPUs.

The whole set of simulations (level 2) to analyze the friction coefficient has been distributed on modern Multi-GPU clusters, which provide CPU-cores and GPUs which can compute several simulations in parallel. A subset of tracer trajectories can be computed in parallel on the CPU-cores and GPUs of a cluster. This way, every CPU-core (GPU) can execute the sequential code in Fortran (CUDA) to compute a single tracer trajectory, and the whole set of tasks can be run on the heterogeneous cluster with the collaboration of the CPU-cores and the GPUs. Moreover, the computational loads of the corresponding tasks are also different because the computation of trajectories in systems with different sizes are needed. Consequently, it is necessary to define an appropriated tasks scheduling to obtain the optimal parallel performance. Several strategies have been devised for this purpose, some of them specifically for particle systems [35]. Here, we have adapted a genetic algorithm (GA) to optimize the trajectories scheduling.

There is a wide variety of previous work where genetic algorithms are used to solve scheduling problems [36]. GA works with a set of individuals which represent every possible solution of the scheduling policy problem (*population*). The procedure evolves iteratively starting with a random set of individuals, P_0, and at every iteration, i, the selection and genetic operators are applied to the population, P_i. Thus, the population is constantly evolving. The selection mechanism allows that the individuals of new populations are closer to the optimal.

The methodology to execute the microrheology model with several system sizes on a Multi-GPU cluster includes the following stages:

1. Profiling stage, which estimates the sequential runtime of the microrheology model on every computational resource (GPU/ CPU-core) for the considered system sizes of the problem.
2. GA scheduling estimation, which plans the set of trajectories on every CPU-core and GPU to optimize the parallel runtime of all simulations by the GA. The inputs of this stage are: the profiling stage output, the number of computational resources of every type on the cluster (number of GPUs/ CPU-cores) and the number of trajectories of every system size of the model.
3. Parallel execution of the model on the cluster according to the scheduling estimation.

The software to carry out stages 2 and 3 has been implemented in Python and is freely available at https://github.com/2forts/GENS.

To analyze the friction coefficient, we study the set of simulations with sizes: $N = 216, 512, 1000, 2197, 4096, 8000, 15625$ and 32768, with 500 trajectories of 500 time units (corresponding to 10^6 time steps). So, the model has to compute a total of 4000 trajectories. A state-of-the-art cluster has been considered as the test platform. It is composed by 4 nodes with a multiprocessor of 16 CPU-cores (Bullx R424-E3 Intel Xeon E5 2650 with 8 GB RAM) and 2 GPUs NVIDIA Tesla M2070.

Table 1 shows the runtime on a CPU-core and a GPU to simulate a single trajectory on such test platform obtained in the profiling stage. The acceleration factors (AF) on GPU vs. CPU-core to simulate a single trajectory are also included. High acceleration factors, specially for large system sizes, are obtained (up to 24×). However, for small problems the use of the GPU computing has no advantage.

From the profiling data, the scheduling is estimated by the GA according to the available resources on the cluster. If the 8 GPUs of the cluster and 56 CPU-cores are used (8 CPU-cores are devoted to controlling the 8 GPUs) to simulate the 4000 trajectories included in the model, the runtime is 2024 h when the GA scheduling is applied. If only a multiprocessor of a node was exploited, the runtime would be 2905 h for all trajectories. To illustrate the advantages of the GA scheduling in terms of runtime for the analyzed model, Table 2 shows the runtime for the GA and a Round Robin approach for several configurations of the cluster and also the

Table 1
Execution time (in seconds) of the simulation of a single trajectory for the eight sizes of the problem (N). t_{GPU} and t_{CPU} columns identify the runtime for a single trajectory on a GPU NVIDIA Tesla M2070/a and CPU-core Bullx R424-E3, respectively. *AF* is the acceleration factor of a GPU vs. a CPU-core for each N.

N	t_{GPU}	t_{CPU}	*AF*
216	1580	790	0.5
512	1785	1860	1.0
1000	2240	3715	1.7
2197	2930	8710	3.0
4096	4450	18065	4.1
8000	7650	43080	5.6
15625	12050	113940	9.5
32768	20012	479313	24.0

Table 2
Parallel execution time, in hours, for a Round Robin placement (RR) and the GA solving 500 trajectories of 500 time units (N = 216, 512, 1000, 2197, 4096, 8000 and 15625) for the cases A) 14 CPU-cores and 2 GPUs; B) 28 CPU-cores and 4 GPUs; C) 28 CPU-cores and 8 GPUs; D) 56 CPU-cores and 8 GPUs; E) 8 GPUs and F) 64 CPU-cores. GAAF shows the GA acceleration factor.

	A	B	C	D	E	F
RR	844.8	422.4	369.6	211.2	290.4	211.2
GA	410.4	206.4	139.2	102.5	283.2	206.4
GAAF	2.1	2.0	2.7	2.1	1.0	1.0

GA acceleration factor (GAAF). The results show that the GAAF ranges from $1,0\times$ (the homogeneous cluster configurations) to $2,7\times$ (the most heterogeneous cluster configuration). So the more heterogeneous the cluster is, the more advantages the GA reaches. Therefore, the use of a multi GPU–CPU cluster in combination with the GA scheduling has allowed that every simulation is executed on a CPU/GPU according to its size, reaching a considerable reduction of the total runtime of the microrheology model.

4. Results

The effective friction coefficient probed by a tracer of the same size as the bath particles develops a plateau for small driving forces, and decreases for increasing forces [7], in analogy with the shear viscosity in bulk systems. We will focus here on the linear regime at small forces, with large tracers, which is expected to be closer to the model of Newtonian fluids used in the theoretical description. Thus, we study first the behavior of γ_{eff} with the external force for a tracer three times larger than the bath particles, $a_t/a = 3$. Fig. 2 shows the results for two systems with different number of particles: $N = 2197$ and $N = 216$. Indeed, γ_{eff} depends strongly on the system size, as expected from the theoretical analysis discussed previously.

The effective friction coefficient shows the same qualitative behavior for a large tracer as the previously reported for $a_t = a$ (a plateau at small forces, followed by a force-thinning regime), irrespective of the number of particles in the system. The linear regime at small forces extends to $F_{\text{ext}} \approx 10\, k_BT/a$. Thus, we select a force of $F_{\text{ext}} = 2.5\, k_BT/a$, which is well inside this linear regime. In the following, we first study finite size effects in different cases, and then compare with the theoretical model.

The first point we analyze is the volume occupied by the tracer. In the simulations shown in Fig. 2, the tracer is inserted in the system, compressing it, and increasing effectively the volume fraction of the bath. Note that the increase of volume fraction is larger for smaller systems. In the theoretical model, the properties of the bath are not affected by the insertion of the tracer, or by modifying the tracer lattice spacing, namely, the simulation box size. We

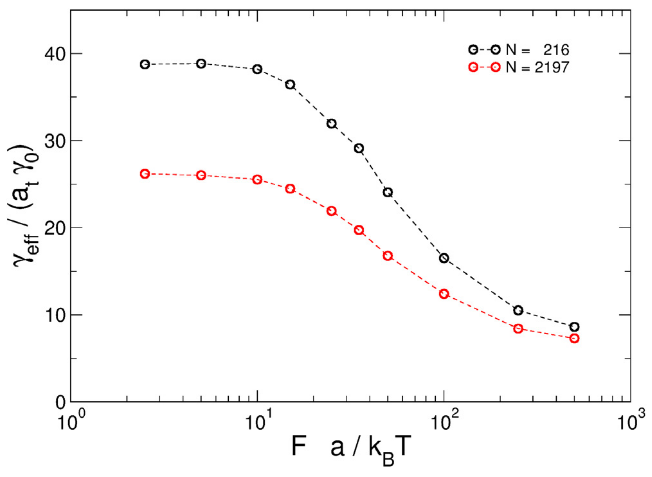


Fig. 2. Effective friction coefficient for a system with $a_t/a = 3$, normalized with the friction with the solvent $a_t\gamma_0$. Two system sizes are considered, $N = 216$ and $N = 2197$, as labeled.

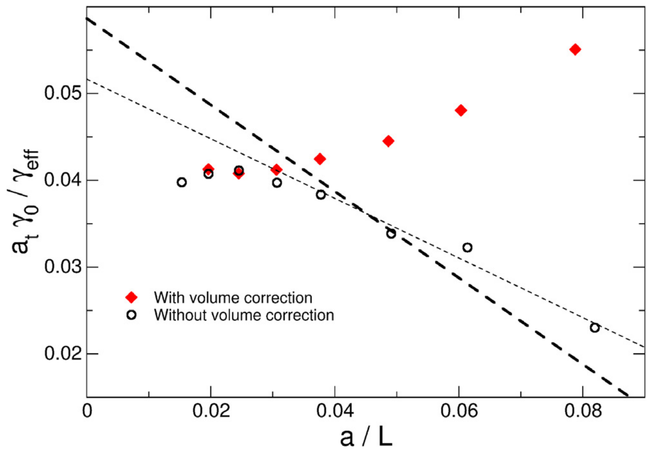


Fig. 3. Effect of considering the volume of the tracer: normalized inverse effective friction coefficient for a system with $a_t/a = 3$ as a function of the inverse system size, for a system with volume correction, or without it, as labeled. The thick dashed line is a free linear fitting to the data without volume correction, for systems up to $N = 8000$ particles, and the thick line is the fitting according to the theoretical model, Eq. (3).

have thus run simulations with different simulation boxes keeping the volume fraction of the system (bath and tracer) constant, and equal to 50% in all cases. The results are presented in Fig. 3, in comparison with the data from simulations without correcting the tracer volume. Both data sets agree for large systems, where the volume of the tracer is negligible, compared with the volume of the whole system, but differ significantly for small systems. Notably, in this case, the friction coefficient is much larger when the tracer volume is not added to the system, due to the increase of effective volume fraction in the bath (note that the inverse friction coefficient is plotted in the figure).

In comparing with the theoretical prediction, we note that the simulation data for the system without the volume correction fits better to the expected qualitative behavior for small systems (namely, a decreasing linear trend of $1/\gamma_{\text{eff}}$ vs. $1/L$). However, the results from the theoretical model are more strict than a linear dependence of $1/\gamma_{\text{eff}}$ vs. $1/L$; Eq. (3) implies a relation between the slope and intercept. The thick dashed line in Fig. 3 shows the theoretical prediction (γ_∞ is fitted), for small and intermediate systems. The fitting is not satisfactory, probably due to the important differences between the simulated system and the theoretical

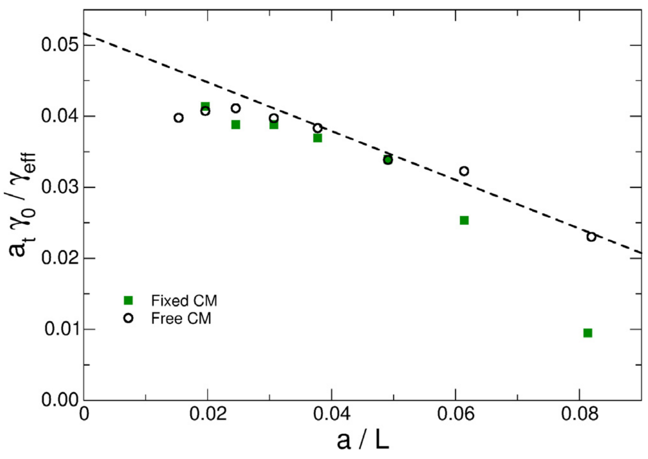


Fig. 4. Effect of fixing the center of mass of the system in microrheology: normalized inverse effective friction coefficient as a function of the inverse system size, for a system with the CM fixed or free, as labeled. The dashed line is a linear fitting to the data for small and intermediate systems.

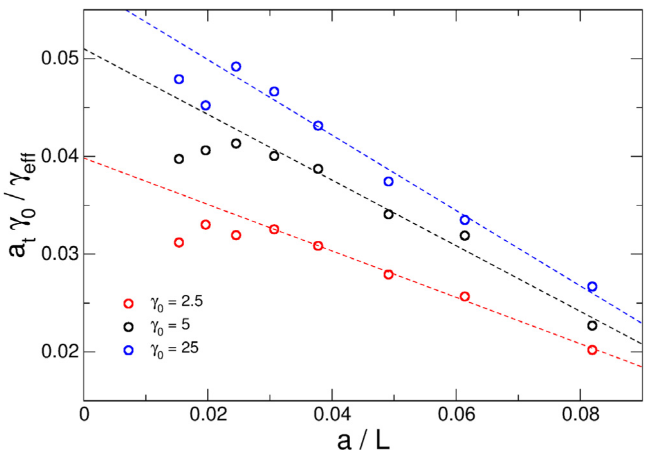


Fig. 5. Effect of the solvent friction coefficient γ_0 on the effective friction coefficient. The lines show the linear fittings (dashed lines).

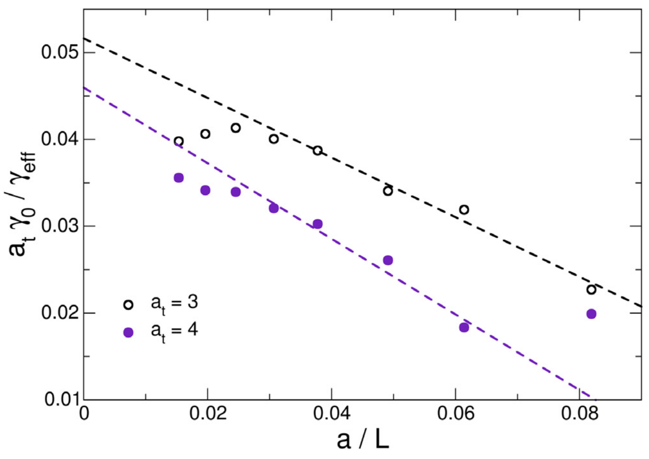


Fig. 6. Effect of varying the tracer size, as labeled. The dashed lines show again the linear fittings to the data from small and intermediate systems.

model (continuous bath vs. particles, Newtonian fluid vs. viscoelastic bath, boundary condition in the tracer surface...).

It must be mentioned that these deviations were not observed in passive microrheology [16,17] (recall that Newtonian dynamics were used those simulations and no force was applied). In our case, however, the agreement with the theory is not quantitative. Still, a decreasing linear dependence is found for $1/\gamma_{\rm eff}$ vs. $1/L$ (thin dashed line) although for large systems the data deviate from the linear trend.

The effect of fixing the center of mass (CM) of the system, or leaving it free is studied next. The application of an external force, even though applied onto a single particle, implies a displacement of the CM, while in a macroscopic system, this should be fixed. The CM can be fixed in the simulations artificially, correcting the particle positions at every time step. Again, this aspect is absent in the theoretical model, where the solvent is an incompressible Newtonian fluid. Fig. 4 presents the results of $\gamma_{\rm eff}$ for different system sizes from simulations with the CM fixed, in comparison with the results leaving it free (in both cases the tracer volume has not been considered in the total volume of the system).

The results for both cases differ for small sizes, but agree for large systems, when the effect of pulling a single particle becomes negligible, and thus the correction of the particle motion is less important. The data with the CM fixed deviate from a linear trend at both small and large systems, while the data for the free CM follows the linear trend, as shown above. Thus the following simulations are run with the CM free, to improve the comparison with the theory.

The microscopic motion of the particles is also affected by the solvent friction coefficient, $\gamma_i = \gamma_0 a_i$ – large values of γ_0 reduce the effect of inertia, making the dynamics more Brownian like. Again, the theoretical model does not consider explicitly this aspect; the bath in the theory is a continuous medium and not Brownian. Fig. 5 presents the results of simulations with different values of γ_0. Upon increasing the friction with the solvent, $\gamma_{\rm eff}/\gamma_0$ decreases, as the contribution from the solvent to the total friction experienced by the tracer increases. On the other hand, in all cases, the inverse friction coefficient shows a linear dependence on $1/L$ for not-too-large systems, as shown by the thin dashed lines.

Finally, we study the effect of the tracer size, as the assumption of a continuous bath is expected to be applicable if the tracer is much larger than the bath particles. Fig. 6 presents the results of the inverse effective friction coefficient as a function of the inverse system size for $a_t/a = 3$ and $a_t/a = 4$. In both cases, the CM is free, the volume of the tracer has not been considered, and $\gamma_0 = 5\sqrt{mk_BT}/a$. The results show the same trend for both sets of data; namely, a linear trend appears for small systems (dashed lines), while both of them deviate for large systems. The figure also allows checking if $\gamma_{\rm eff}$ is linear with the tracer size, as expected from Stokes' law. The results, however, indicate that the friction coefficient grows with a_t faster than linear, indicating that the Stokes' regime is not valid in this system and for this range of tracer sizes.

The analysis presented so far have shown that the theoretical prediction is only qualitatively followed for small and intermediate systems, with the appropriate conditions in the simulations, but fails for large systems in all cases. Thus, we analyze in the following the reason for this discrepancy.

Within the theoretical model, the friction experienced by the tracer is calculated from the velocity profile in the bath [37]. Therefore, we study the velocity of the bath particles in the simulations and compare it with the theoretical result. Because for large systems $\gamma_{\rm eff}$ becomes independent on the system size, we focus on the system with $N = 15625$ particles and compare the velocity map with the theoretical results for a single particle in an incompressible Newtonian fluid. The latter is given, in polar coordinates, by [37]:

$$v_r(r,\theta) = u_{\rm tracer}\cos\theta\left(\frac{3a_t}{2r} - \frac{a_t^3}{2r^3}\right) \qquad (4)$$

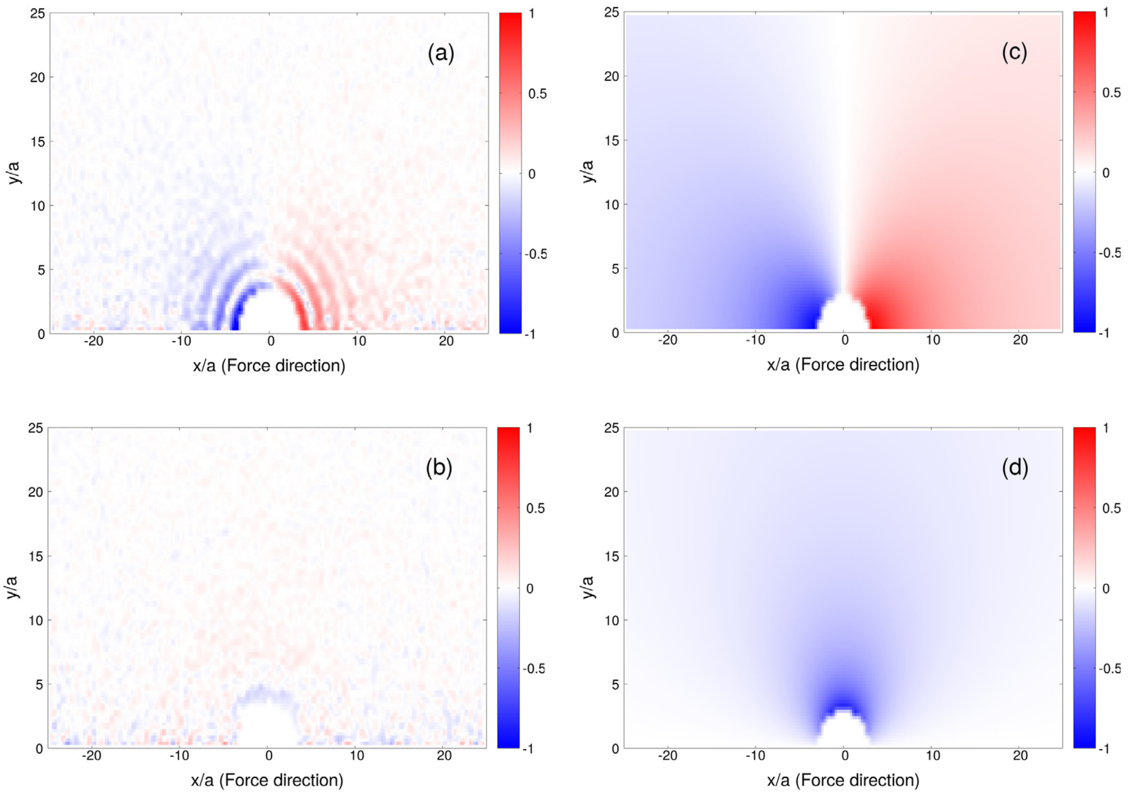


Fig. 7. Velocity maps in the bath from simulations (panels (a) and (b)) and theory (panels (c) and (d)); the radial (angular) component is presented in the top (bottom) panels.

$$v_\theta(r,\theta) = -u_{\text{tracer}} \sin\theta \left(\frac{3a_t}{4r} + \frac{a_t^3}{4r^3}\right) \quad (5)$$

where θ is the angular coordinate, measured from the force direction, and u_{tracer} is the tracer velocity. This result is valid for low Reynolds number, which is indeed our case; a rough estimate of the ratio of inertia to viscous forces is $(mu_{\text{tracer}}/a_t)/(\gamma a_t)$ which is below 0.1 in all cases.

Fig. 7 shows both components of the velocity in the bath (divided by the tracer velocity) from simulations (left panels) and theory (right ones). In the simulations, the system with $N = 15625$ has been run for ca. $5 \cdot 10^4$ time units (amounting to 10^8 integration steps), to improve the statistics. The comparison shows some important differences, and some similarities. The normal component shows that the velocity in front of the tracer is positive in both the simulations and theory, and negative behind it; however while it decays monotonously to zero from the tracer to infinity in the theory , panel (c), in the simulations, it oscillates — panel (a). Transversal to the force, the velocity has a negligible normal component both in the simulations and in the theory.

The angular component (bottom panels), on the other hand, shows also some differences between the simulations and the theory. In the model, panel (d), the minimum is in the tracer surface, perpendicular to the force direction (recall that stick boundary conditions are assumed in the tracer surface), and decays monotonously to zero. The simulations, shown in panel (b), are much noisier, but a dip develops in this region; further from the tracer (in the direction perpendicular to the force), v_θ becomes positive (but small) and decays to zero far from the tracer. In this case, oscillations are not observed, probably due to the poor statistics.

The radial component is studied in more detail in Fig. 8, where the bath velocity in front of the tracer is studied (to reduce the

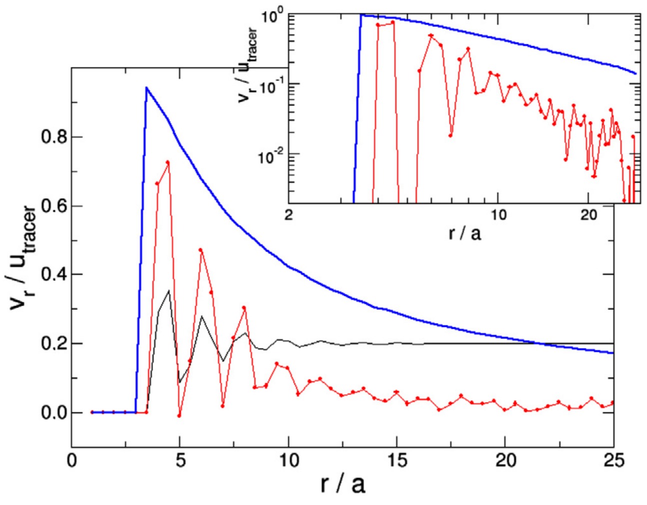


Fig. 8. Normalized radial component of the bath velocity in front of the tracer, from simulations (red points and lines) and theory (thick blue line). The bath density is also presented (thin black line), scaled. The inset shows the same velocity data in logarithmic scale. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

statistical noise, it has been averaged over an angle of 30°). The oscillations of the bath velocity have a wavelength of one (bath) particle diameter, and reproduce the oscillations of the density profile (black line). Thus, these are a direct consequence of the finite size of the bath particles. The inset to this figure presents the same data on a logarithmic scale, showing that the bath velocity in front of the particle decays faster in the simulations than in the theory, probably due to the friction with the solvent and the Brownian motion that disrupt the transfer of momentum in the bath. Notably, the velocity in the bath is negligible, within the noise level, for distances of ca. $20 - 25a$. The systems with a box size smaller than (twice) this value should present *interactions* between the tracer and its periodic images due to the cutoff of the velocity profile. In systems with a large simulation box, the velocity profile can decay inside the box, and the friction experienced by the tracer is not affected by its periodic images. It can be concluded, therefore, that large systems, beyond this limit, should be free of finite size effects, and the limit corresponds to the systems of 8000 particles, as indeed observed in the evolution of γ_{eff} vs. L, studied previously.

5. Conclusions

We have presented simulations of active microrheology in hard colloids. This system is strongly affected by finite size effects, which have been analyzed using a model developed previously for an array of tracers moving in a Newtonian fluid. According to the model, the inverse effective friction felt by the tracer depends linearly on the inverse lattice parameter. In order to test this prediction, different simulation conditions have been proposed (consideration of the tracer volume, motion of the center of mass, tracer size, and friction with the solvent), but deviations for large systems are observed in all cases, although linear trends have been observed in particular cases for small and intermediate systems.

The deviations of γ_{eff} with respect to the theoretical behavior indicate that the approximations in the theoretical model are too strong for the simulated system. This invalidates the use of this model to extract the effective friction coefficient for a macroscopic system from the linear extrapolation $1/L \rightarrow 0$. Previous simulations of passive microrheology in a bath of particles undergoing molecular dynamics did follow the linear trend, and could use it to extract γ_∞ and check Stokes' law. The failure of the model in our case must be attributed to the different microscopic dynamics, which is dissipative in our case. This, presumably, implies a damping for the shear waves, and a saturation of the finite size effect predicted by the theory.

Computationally, on the other hand, it presents an important advantage; namely, simulations in a large enough system can be used to obtain the effective friction coefficient probed by a tracer, skipping the necessity of making simulations with different system sizes and rely on an extrapolation to $1/L \rightarrow 0$. It must be mentioned, however, that the minimum size of the system to avoid finite size effects may depend on the tracer size.

The results presented here concern a system of quasi-hard spheres, i.e. with short-range interactions. While it is difficult to extrapolate these results for systems with interactions of longer range, structural correlations will extend to larger distances, possibly increasing the range of validity of the continuous medium approximation for the bath. However, if the microscopic dynamics is indeed responsible for the damping of the shear waves and hydrodynamic correlations, as proposed here, it should eventually dominate over long enough distances, making a proper analysis of finite size effects mandatory. The technique proposed here (GA scheduling of simulation tasks of different loads) is a powerful tool for this analysis.

Acknowledgments

This work has been supported by the Spanish Science and Technology Commission (CICYT) under contracts TIN2015-66680 and FIS2015-69022-P; Junta de Andalucía under contract P12-TIC-301 in part financed by the European Regional Development Fund (ERDF). G. Ortega is a fellow of the Spanish 'Juan de la Cierva Incorporación' program.

References

[1] P.M. Chaikin, T.C. Lubensky, Principles of Condensed Matter Physics, Cambridge University Press, Cambridge (UK), 1995.
[2] A. Fernandez-Nieves, A.M. Puertas (Eds.), Fluids, Colloids and Soft Materials, John Wiley & sons, New Jersey (USA), 2016.
[3] R. Khare, D.S. Devarajan, Curr. Opin. Chem. Eng. 16 (2017) 86.
[4] R.G. Larson, The Structure and Rheology of Complex Fluids, Oxford University Press, USA, 1999.
[5] P. Cicuta, A.M. Donald, Soft Matter 3 (2007) 1449–1455.
[6] L. Wilson, W.C.K. Poon, Phys. Chem. Chem. Phys. 13 (2011) 10617–10630.
[7] A.M. Puertas, Th. Voigtmann, J. Phys.: Condens. Matter 26 (2014) 243101.
[8] E. Furst, T. Squires, Microrheology, Oxford University Press, UK, 2017.
[9] T.G. Mason, D.A. Weitz, Phys. Rev. Lett. 74 (1995) 1250.
[10] T.G. Mason, H. Gang, D.A. Weitz, J. Mol. Struct. 383 (1996) 81–90.
[11] M. Umansky, D. Weihs, Comput. Phys. Comm. 183 (2012) 1783–1792.
[12] H.-H. Jeong, A.G. Mark, T.-C. Lee, M. Alarco-Correa, S. Eslami, T. Qiu, J.G. Gibbs, P. Fischer, Nano Lett. 16 (2012) 4887–4894.
[13] T.M. Squires, Langmuir 24 (2008) 1147–1159.
[14] T.M. Squires, T.G. Mason, Annu. Rev. Fluid Mech. 42 (2010) 413.
[15] H. Hasimoto, J. Fluid Mech. 5 (1959) 317–328.
[16] I.-C. Yeh, G. Hummer, J. Phys. Chem. B 108 (2004) 15873–15879.
[17] R.O. Sokolovskii, M. Thachuk, G.N. Patey, J. Chem. Phys. 125 (2006) 204502.
[18] F. Ould-Kaddour, D. Levesque, Phys. Rev. E 63 (2000) 011205.
[19] J.R. Schmidt, J.L. Skinner, J. Chem. Phys. 119 (2003) 8062–8068.
[20] H.K. Shin, C. Kim, P. Talkner, E.K. Lee, Chem. Phys. 375 (2010) 316–326.
[21] I.C. Carpen, J.F. Brady, J. Rheol. 49 (2005) 1483–1502.
[22] I. Gazuz, A.M. Puertas, Th. Voigtmann, M. Fuchs, Phys. Rev. Lett. 102 (2009) 248302.
[23] C.J. Olson Reichhardt, C. Reichhardt, Phys. Rev. E 82 (2010) 051306.
[24] M.V. Gnann, I. Gazuz, A.M. Puertas, M. Fuchs, Th. Voigtmann, Soft Matter 7 (2011) 1390.
[25] Ch.J. Harrer, D. Winter, J. Horbach, M. Fuchs, Th. Voigtmann, J. Phys.: Condens. Matter. 24 (2012) 464105.
[26] D. Winter, J. Horbach, P. Virnau, K. Binder, Phys. Rev. Lett. 108 (2012) 028303.
[27] D. Winter, J. Horbach, J. Chem. Phys. 138 (2013) 12A512.
[28] T.M. Squires, J.F. Brady, Phys. Fluids 17 (2005) 073101.
[29] G. Ortega, A.M. Puertas, E.M. Garzón, J. Supercomput. 73 (2017) 370–383.
[30] J.K.G. Dhont, An Introduction to Dynamics of Colloids, Elsevier, Amsterdam (Holland), 1996.
[31] E. Lange, J.B. Caballero, A.M. Puertas, M. Fuchs, J. Chem. Phys. 130 (2009) 174903.
[32] W. Paul, D.Y. Yoon, Phys. Rev. E 52 (1995) 2076.
[33] G. Ortega, A.M. Puertas, F.J. de Las Nieves, E.M. Garzón, Algorithms and Architectures for Parallel Processing: 16th ICA3PP, Granada, Spain, Springer International Publishing, Cham, 2016, pp. 457–466.
[34] I.V. Morozov, A.M. Kazennov, R.G. Bystryi, G.E. Norman, V.V. Pisarev, V.V. Stegailov, Comput. Phys. Comm. 182 (2011) 1974–1978.
[35] W.M. Brown, P. Wang, S.J. Plimpton, A.N. Tharrington, Comput. Phys. Comm. 182 (2011) 898–911.
[36] V. Sels, J. Coelho, A. Dias, M. Vanhoucke, Comput. Oper. Res. 53 (2015) 107–117.
[37] E. Guyon, J.P. Hulin, L. Petit, C.D. Mitescu, Physical Hydrodynamics, Oxford University Press, Oxford (UK), 2001.

2.2.2 Dynamics and friction of a large colloidal particle in a bath of hard spheres: Langevin dynamics simulations and hydrodynamic description

Title	Dynamics and friction of a large colloidal particle in a bath of hard spheres: Langevin dynamics simulations and hydrodynamic description		
Authors	*F. Orts, G. Ortega, E.M. Garzón, M. Fuchs and A.M. Puertas*		
Journal	Physical Review E		
Year	2020		
Volume	101 (052607)		
Pages	1-10		
DOI	http://dx.doi.org/10.1103/PhysRevE.101.052607		
IF (JCR 2019)	2.296		
Categories	Physics, Mathematical:	9/55	**(Q1)**
	Physics, Mathematical:	13/34	(Q2)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, has helped with the computation associated to this paper.

Dynamics and friction of a large colloidal particle in a bath of hard spheres: Langevin dynamics simulations and hydrodynamic description

F. Orts,[1] G. Ortega,[1] E. M. Garzón,[1] M. Fuchs,[2] and A. M. Puertas[3]
[1]*Departamento de Informática, Campus de Excelencia Internacional Agroalimentario (ceiA3), Universidad de Almería, 04120 Almería, Spain*
[2]*Fachbereich Physik, Universität Konstanz, 78457 Konstanz, Germany*
[3]*Departamento de Física Aplicada, Universidad de Almería, 04120 Almería, Spain*

(Received 25 July 2019; revised manuscript received 27 March 2020; accepted 29 April 2020; published 18 May 2020)

The analysis of the dynamics of tracer particles in a complex bath can provide valuable information about the microscopic behavior of the bath. In this work, we study the dynamics of a forced tracer in a colloidal bath by means of Langevin dynamics simulations and a theory model within continuum mechanics. In the simulations, the bath is comprised of quasihard spheres with a volume fraction of 50% immersed in a featureless quiescent solvent, and the tracer is pulled with a constant small force (within the linear regime). The theoretical analysis is based on the Navier-Stokes equation, where a term proportional to the velocity arises from coarse-graining the friction of the colloidal particles with the solvent. As a result, the final equation is similar to the Brinkman model, although the interpretation is different. A length scale appears in the model, k_0^{-1}, where the transverse momentum transport crosses over to friction with the solvent. The effective friction coefficient experienced by the tracer grows with the tracer size faster than the prediction from Stokes's law. Additionally, the velocity profiles in the bath decay faster than in a Newtonian fluid. The comparison between simulations and theory points to a boundary condition of effective partial slip at the tracer surface. We also study the fluctuations in the tracer position, showing that it reaches diffusion at long times, with a subdiffusive regime at intermediate times. The diffusion coefficient, obtained from the long-time slope of the mean-squared displacement, fulfills the Stokes-Einstein relation with the friction coefficient calculated from the steady tracer velocity, confirming the validity of the linear response formalism.

DOI: 10.1103/PhysRevE.101.052607

I. INTRODUCTION

In soft matter, different time- and length scales are involved, due to the presence, typically, of simple solvents and macromolecules. This is usually tackled by integrating out the fastest degrees of freedom, which leaves an equation of motion for the relevant (macromolecular) ones [1–3]. A clear example is the Langevin equation for the Brownian motion of a colloidal particle, where the solvent is modelled only through the friction and random forces acting on the particle. This allows the calculation of parameters characterizing the solvent by studying the diffusion of a single particle. This idea has been elaborated further to study more complex fluids and is the core of so-called microrheology.

In microrheology, a single colloidal tracer (or a very small number of them) is introduced in a complex fluid to study its mechanical behavior at the microscopic scale [4–8]. The tracer can be left undisturbed to undergo diffusion in the complex bath due to thermal and density fluctuations (passive microrheology) or forced to probe the response of the bath (active microrheology). Experiments [9–11] and simulations of active microrheology [12–17] have shown that the effective friction coefficient shows a linear dependence on the force for small forces, allowing the definition of a microviscosity. A nonlinear regime is entered for larger forces and a second linear regime, featuring a smaller viscosity, may be attained for large forces. This overall phenomenology resembles that of conventional (bulk) rheology, showing shear thinning, thickening, or more complex scenarios [18–20]. Different possibilites have also been reported in microrheology, depending on the interactions considered [17].

The interpretation of the results from microrheology must take into account all the degrees of freedom. While in dilute cases, theory achieves to consider the bath particles explicitly (e.g., by the direct interactions between the tracer and bath particles, or among the bath particles) [7,21–26], a dense fluid is often described within hydrodynamics. This implies that not only the solvent but also the bath must be treated as continuum fluid [18]. While the solvent is typically a Newtonian fluid, the bath is a complex one, namely the transport coefficients depend on the driving. The models used in microrheology, thus, must describe the interaction of the tracer with these two baths, either as fluids with different properties [27–30], sacrificing a detailed structural description, or using a microscopic theory to describe the motion of the tracer and bath particles in a solvent [31–33].

In this work, we study the dynamics of a large tracer in a dense bath of colloidal particles; the tracer is subjected to an external constant force, small enough to remain in the linear regime. All particles exhibit Langevin motion characterized by a constant friction coefficient with the solvent, which also provides the random forces. They are taken to be Gaussian and white, and the fluctuation dissipation relation holds. This widely used model focuses on the collective interactions among bath particles and tracer, while it neglects the solvent flow, which leads to hydrodynamic interactions [2]. We have run simulations with a tracer up to eight times larger than the bath particles, and a bath volume fraction of $\phi = 0.50$. The results are analyzed using a hydrodynamic model, within the formalism of continuum mechanics. It differs from the (naively expected) Navier-Stokes hydrodynamics even in the limit of macroscopic tracers. The model has been derived coarse-graining systems of Langevin particles [34], and the resulting hydrodynamic equation is the Brinkman equation, which has been applied previously to diffusion in porous systems [35], although our interpretation is different from previous ones. Notably, the solution of the Brinkman equation brings out a length scale where transverse momentum transport crosses over to friction with the solvent. The friction coefficient thus grows with the tracer size much faster than Stokes's law while the velocity profile in the bath decays as the inverse cubed distance to the tracer. After performing a finite-size analysis in the simulation results, the friction coefficient and velocity profile can be correctly rationalized within the theoretical model. The effect of the different boundary conditions on the tracer surface is also discussed. Finally, we study the dynamics of the tracer using the mean-squared displacement and confirm the validity of the Stokes-Einstein relation for all tracer sizes.

II. MODEL

The system we aim to describe is a colloidal bath at high density with a (colloidal) tracer particle equal or larger than the bath particles. There are, therefore, three components in the system: solvent, bath particles, and tracer particle. While the system is in equilibrium, at time $t = 0$ a constant external force starts to pull the tracer. Similar systems have been considered to study microrheology both in simulations [16,23,32,36–38] and in theory [14,21,22,39]. In our case, the force is small enough to drive the system out of equilibrium within the linear regime.

We approach this system from two points of view: using Langevin dynamics simulations and a theoretical model based on continuum mechanics. In both cases, the solvent is assumed to be at rest, its only effect being a friction force proportional to the particles velocities, and a random force which produces Brownian motion. This implies that we neglect hydrodynamic interactions (HI) among all particles but allows us to run simulations of large systems and proceed analytically in the theory, and connects with many previous works where HI are also neglected. This may seem a harsh approximation but its effect on the local cageing of particles is only quantitative [17,40], not affecting the physical behavior of the system, in particular at the high bath density studied here.

A. Simulations

In the simulations, the system under study is composed of N polydisperse particles, including a tracer (labeled with $j = 1$), in a cubic box with periodic boundary conditions. All particles undergo Brownian motion, which we model by the Langevin equation [2]. For particle j, the equation of motion reads:

$$m_j \frac{d^2\,\mathbf{r}_j}{dt^2} = \sum_{i\neq j} \mathbf{F}_{ij} - \gamma_j \frac{d\,\mathbf{r}_j}{dt} + \mathbf{f}_j(t) + \mathbf{F}_{\rm ext}\delta_{j1}, \qquad (1)$$

where m_j is the particle mass; $\mathbf{F}_{ij}$ is the interaction force between particles i and j; γ_j is the friction coefficient with the solvent, assumed to be proportional to the particle radius a_j, $\gamma_j = \gamma_0 a_j$, mimicking Stokes's law; and $\mathbf{f}_j$ is the Brownian force. The latter is random, but its intensity is linked to the friction force, as given by the fluctuation-dissipation theorem, $\langle \mathbf{f}_j(t)\cdot\mathbf{f}_j(t')\rangle = 6k_BT\gamma_j\delta(t-t')$, where k_BT is the thermal energy and $\delta(x)$ is the Dirac-delta symbol [2]. Finally, the external force, $\mathbf{F}_{\rm ext}$, acts only on the tracer (as shown by the Kronecker δ symbol, δ_{j1}). The energy injected by this force is dissipated by the friction of the tracer with the solvent and the bath particles, keeping the kinetic temperature constant in the stationary state. As mentioned above, hydrodynamic interactions have been neglected in the equation of motion.

The interaction potential between particles i and j is derived from the central inverse-power potential:

$$V(\mathbf{r}) = k_BT\left(\frac{r}{a_{ij}}\right)^{-36} \qquad (2)$$

with $r = |\mathbf{r}|$ the center-to-center distance between the particles and $a_{ij} = a_i + a_j$. Due to the high value of the exponent, this system behaves as colloidal hard spheres [41]. To avoid crystallization at high density, a continuous size distribution of width $2\delta = 0.2a$, with a the mean radius, is used for the bath particles. The tracer has radius $a_t \geqslant a$. For the sake of simplicity in the numerical algorithm, all particles, including the tracer, have the same mass: $m_j = m$ (note that the tracer particle gives a scale for the external force). The mean bath particle radius a, the thermal energy k_BT, and particle mass m are the length, energy, and mass units, respectively. The friction coefficient with the solvent of particle j is calculated with $\gamma_0 = 5\sqrt{mk_BT}/a$, which gives a single-particle diffusion coefficient of $D_0 = k_BT/\gamma_0 = 0.2\,a\sqrt{k_BT/m}$ for the mean particle. The Langevin equations of motion are integrated using the Heun algorithm [42], with a time step of $\delta t = 0.0005\,a\sqrt{m/k_BT}$.

In our simulations, the system containing the tracer is equilibrated without external force. For $t > 0$, the constant external force is applied onto the tracer in the z direction, and its trajectory is monitored. The effective friction coefficient experienced by the tracer is obtained from its long-time steady velocity, $\langle v\rangle$, averaged over many independent trajectories, and using the steady-state relationship $F_{\rm ext} = \gamma_{\rm eff}\langle v\rangle$. For small forces, the tracer velocity presents a linear regime with the external force, resulting in a constant friction coefficient, followed by a decrease of $\gamma_{\rm eff}$ for larger forces (nonlinear response) [43]. We focus here on the linear regime at small forces. The connection to the hydrodynamic calculation of the theoretical section below is then given by Onsager's regression hypothesis [8].

In hydrodynamics, it is well known that there are long-range correlations in the fluid, decaying typically with the inverse distance. Although our model predicts a faster decay of these correlations, as shown below, it is mandatory

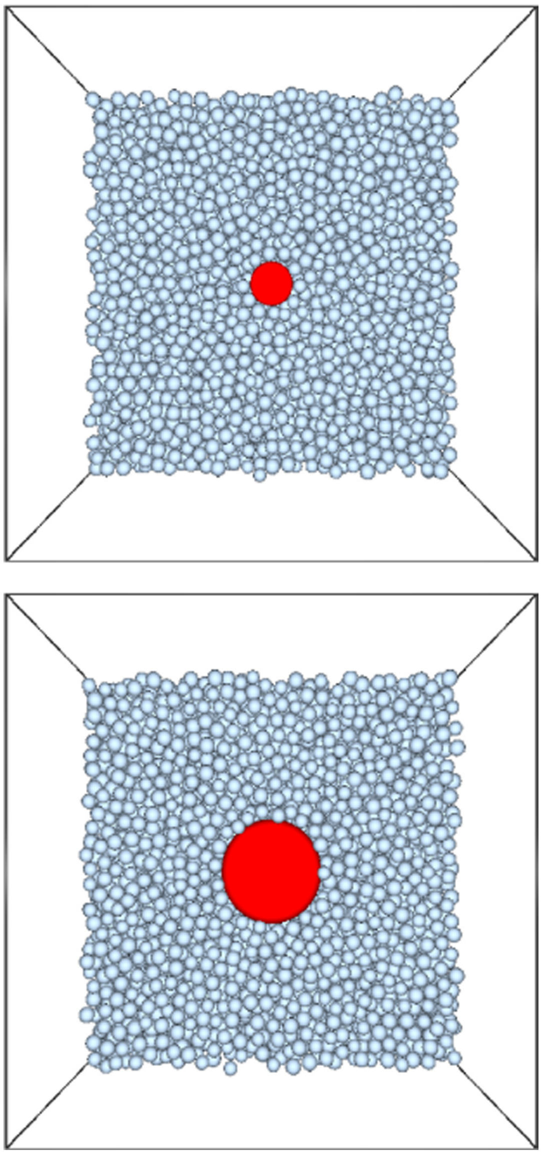


FIG. 1. Snapshots of the systems with $N = 15\,625$, with a tracer with $a_t = 3a$ (top panel) and $a_t = 7a$ (bottom panel). The tracer is marked in red, and the particles in front of it have been removed for clarity.

to perform an analysis of finite-size effects. In fact, since periodic boundary conditions are used, an infinite cubic array dragged through a bath of particles is considered. We have thus run simulations of systems with $N = 216$, 512, 1000, 2197, 4096, 8000, 15 625, and 32 768 particles and tracer sizes from $a_t = a$ to $a_t = 8a$. Figure 1 presents two snapshots of the system with $N = 15\,625$ particles, including a tracer of size $a_t = 3a$ [Fig. 1 (top)] and $a_t = 7a$ [Fig. 1 (bottom)]. The bath volume fraction is $\phi = 0.50$ in all cases, and the volume occupied by the tracer is not accounted for in the calculation of the simulation box size [43]. The center of mass of the system is not fixed when the external force is applied. For the calculation of the friction coefficient, 500 tracer trajectories have been analyzed. In addition to the tracer dynamics, the density and velocity profile in the bath have been studied in several cases to check the theoretical predictions; note that Langevin dynamics gives directly the particle velocity.

Numerical implementation

From a computational point of view, the requirement of a finite-size analysis implies running simulations with different number of particles, N, for every tracer size, a_t. For this purpose we have used high performance computing in two ways: (i) programming in graphics processing units (GPU) to speed up the simulation of a single trajectory, and (ii) using a genetic algorithm (GA) to balance the load of all the processing units of the computer cluster, taking into account the different durations of the simulations with different N.

We have accelerated the computation of a single tracer trajectory by means of GPU computing using the CUDA interface [43–45]. Note that the full system with N particles has to be simulated, although the trajectory of a single particle (the tracer) is the most relevant. In particular, the calculation of the interaction forces among all particles and the integration of the equations of motion are very demanding, and have been thoroughly optimized [46]. In addition to this CUDA-GPU core, a standard sequencial FORTRAN code has been used in the CPUs. It was checked that both codes give the same results when the same sequence of random numbers is used for the Brownian force.

The whole set of simulations to analyze the friction coefficient has been run on modern multi-GPU clusters, that provide CPU cores and GPUs which can compute several simulations in parallel. Since the simulations of systems with different sizes are needed, the computational loads of the corresponding tasks are also different. Therefore, an appropriate balance for the execution is decisive. Here we have adapted a genetic algorithm to achive the optimal parallel performance [47]. In our GA, a set of possible solutions of the scheduling problem is the *population*. The algorithm evolves iteratively, starting with a random population, using the mutation and selection mechanisms until the optimal solution is reached, as defined by the minimum spread in execution times among all processing units. A code written in Python has been developed to calculate the optimal distribution of tasks.

In our procedure, a single trajectory in every unit (CPU core or GPU core) is executed for every size and a given tracer radius, and the running times are recorded. With these times, the optimal distribution of trajectories per unit is calculated, ensuring that all units finish their tasks with a minimum difference. This distribution is then passed to the cluster to perform the whole set of simulations for a single tracer size. As mentioned above, simulations with $N = 216, 512, 1000, 2197, 4096, 8000, 15\,625$, and $32\,768$ particles have been run, ensuring that all particles can fit into the simulation box (recall that the tracer volume is not accounted for in the calculation of the simulation box size). Thus, for large tracers, only the biggest systems are simulated. Every trajectory has been recorded for 10^6 time steps, corresponding to $t = 500\,a\sqrt{m/k_BT}$ or $t = 100a^2/D_0$. This time is long enough to reach the stationary state and provide a correct estimation of the tracer velocity, as checked with longer simulations in selected cases.

A cluster composed by four nodes with a multiprocessor of 16 CPU cores (Bullx R424-E3 Intel Xeon E5 2650 with

TABLE I. Runtime in seconds of the simulation of a single trajectory for different sizes (N). The t_{GPU} and t_{CPU} columns show the execution time for a single trajectory on a GPU NVIDIA Tesla M2070/a and CPU-core Bullx R424-E3, respectively.

N	t_{GPU}	t_{CPU}
216	1580	790
512	1785	1860
1000	2240	3715
2197	2930	8710
4096	4450	18 065
8000	7650	43 080
15 625	12 050	113 940
32 768	20 012	479 313

8 GB RAM) and 2 GPUs NVIDIA Tesla M2070 has been used. Table I shows the runtime on a CPU core and a GPU to simulate a single trajectory (profiling stage) for the systems with $a_t = 3a$. Note that GPU programming is particularly advantageous for large systems (up to 24× faster), although the sequencial code is faster for small systems.

B. Theory

We now search for a continuum mechanics description, in order to understand the motion of a macroscopic tracer in the bath of interacting Brownian particles. This search is motivated by the success of Stokes's calculation of the friction of a macroscopic tracer in a Newtonian fluid. He obtained it based on the Navier-Stokes equation (NSE) for the velocity of a continuous Newtonian fluid subjected to external stresses or forces. In colloid science, Stokes's law describes an isolated rigid particle immersed in a solvent which is dragged with a constant velocity, with stick (or slip) boundary conditions on the particle surface. The resulting friction force depends linearly on the solvent viscosity and the bead radius and is proportional to its velocity.

Here to describe the tracer in a colloidal bath we have to coarse-grain the system of coupled Langevin equations for the bath particles $j = 2, \dots, N$ in Eq. (1). This was recently performed using the Zwanzig-Mori projection operator technique [48,49] and considering the long-wavelength limit [34]. The presence of the solvent leads to the inclusion of an additional friction term in the NSE, proportional to the bath particle velocity field, $\mathbf{u}$. This accounts for the local dissipation of the bath particles in the solvent, and arises from coarse-graining the drag forces on the particles [34]. In the stationary state, the hydrodynamic equation reads:

$$\nabla P - \eta_0 \nabla^2 \mathbf{u} = -\zeta_0 \mathbf{u} + \mathbf{F}_{\mathrm{ext}}, \tag{3}$$

where P is the pressure. This equation contains the hydrodynamic friction with a bath of viscosity η_0 (that represents the colloidal system), and with an inert solvent, of friction coefficient ζ_0 (representing the solvent) as well as an external force acting on the system. Without hydrodynamic interactions, the friction coefficient in incompressible systems is simply $\zeta_0 = n\gamma_0$ where n is the bath number density. For the calculation of the analog of Stokes's friction, the external force $\mathbf{F}_{\mathrm{ext}}$ is assumed to be a point force acting on the tracer center. This equation is complemented by the incompressibility condition:

$$\nabla \cdot \mathbf{u} = 0. \tag{4}$$

Equation (3) was already proposed by Brinkman to describe the motion of a tracer in a swarm of colloidal particles [35], as a combination of Darcy's equation and the NSE. However, the interpretation of the parameters is different: In the Brinkman model, the divergence of the stress tensor represents the solvent, and the linear term in $\mathbf{u}$ is due to the presence of the other particles, which act as a porous matrix. Tam [50] used a more rigorous derivation to this equation from first principles, albeit with the same interpretation. Due to this interpretation, the Brinkman equation has been widely used to study the diffusion in a porous medium [51]. It must be also mentioned that the Brinkman equation is similar to the Laplace-transformed unsteady Navier-Stokes equation.

It has been shown previously [34] that Eq. (3) holds with or without hydrodynamic interactions. It requires that momentum is not conserved (as holds in the Langevin simulations, where the solvent relaxes the momenta), yet that the bath viscosity η_0 is large in order for a region (later identified by the wave vector k_0) to emerge where the NSE holds in approximation. As any continuum mechanics description, application of Eq. (3) requires smooth and slow fluctuations, which translates into large tracer sizes. As specific approximation, Eq. (3) neglects the diffusive build-up of a density profile around the forced tracer, which could become noticeable in an appreciably compressible system. It is also interesting to note that the Brinkman's equation is not Galilei invariant, which is different from the NSE. This is in agreement with the Langevin equation, which is also not Galilei invariant. On the other hand, this implies that the problem of the moving sphere in a quiescent fluid is different from a fixed sphere in an incoming fluid. The problem we are interested in is the former one, namely, a moving tracer in a quiescent fluid.

This case has been solved previously in the literature, see, e.g., Ref. [52], giving a velocity profile around the tracer (located at $\mathbf{r} = 0$):

$$\mathbf{u}(\mathbf{r}) = \frac{1}{8\pi\eta_0}\mathcal{S}(\mathbf{r}) \cdot \mathbf{F}_s + \mathbf{u}_{\mathrm{hom}}(\mathbf{r}), \tag{5}$$

where $\mathcal{S}(\mathbf{r})$ is a matrix of elements:

$$\mathcal{S}_{ij}(\mathbf{r}) = \delta_{ij}\frac{\mathcal{A}(r)}{r} + \frac{r_i r_j}{r^3}\mathcal{B}(r) \tag{6}$$

with

$$\mathcal{A}(r) = 2\left(1 + \frac{1}{k_0 r} + \frac{1}{k_0^2 r^2}\right)e^{-k_0 r} - \frac{2}{k_0^2 r^2}, \tag{7}$$

$$\mathcal{B}(r) = -2\left(1 + \frac{3}{k_0 r} + \frac{3}{k_0^2 r^2}\right)e^{-k_0 r} + \frac{6}{k_0^2 r^2}, \tag{8}$$

and $\mathbf{F}_s = F_s \hat{e}_z$ is an effective surface force that depends on the boundary conditions (see below). The inverse distance k_0, appearing in the expressions above is defined as $k_0 = \sqrt{\zeta_0/\eta_0}$ and describes the length scale of the crossover from friction at large distances, originating from the coupling of the particles to the solvent according to the Langevin equation, to diffusive transverse momentum transport intrinsic in the NSE based on Newtonian dynamics, for short distances. The ratio between this length scale and the tracer size, viz. the dimensionless parameter $k_0 a_t$, plays a central role in the following results; for $k_0 \to 0$ the NSE description of a particle in a Newtonian solvent is recovered, whereas for $k_0 \to \infty$ the innert solvent is dominant. In particular, for small k_0:

$$\lim_{k_0\to 0} \mathcal{A}(r) = \lim_{k_0\to 0} \mathcal{B}(r) = 1, \qquad (9)$$

which recovers the velocity profile for the Newtonian solvent [53].

The second term in Eq. (5), $\mathbf{u}_{\rm hom}$, is the velocity profile without external force and pressure, which decays exponentially:

$$\mathbf{u}_{\rm hom}(\mathbf{r}) = -\hat{e}_r \frac{F_h a^2 e^{-k_0 r}}{4\pi \eta_0 r^3}(1 + k_0 r)\cos\theta$$

$$+\hat{e}_\theta \frac{F_h a^2 e^{-k_0 r}}{8\pi \eta_0 r^3}\left(1 + k_0 r + k_0^2 r^2\right)\sin\theta. \qquad (10)$$

Here F_h has to be determined by the boundary conditions, as well as F_s. For stick boundary conditions,

$$\mathbf{u}(a_t) = u_0, \quad \text{and} \quad \mathbf{u}(r \to \infty) = 0$$

with u_0 the tracer velocity. This yields:

$$F_s = 6\pi \eta_0 a_t u_0 \left(1 + k_0 a_t + \frac{1}{3}k_0^2 a_t^2\right)$$

and

$$F_h = -4\pi \eta_0 a_t u_0 \left(1 + \frac{3}{k_0 a_t} + \frac{3}{k_0^2 a_t^2} - 3\frac{e^{k_0 a_t}}{k_0^2 a_t^2}\right). \qquad (11)$$

For slip boundary conditions, on the other hand, it is customary to introduce a slip length, b, and replace the condition of the surface velocity with

$$u_r(a_t) = u_0 \cos\theta, \qquad \text{and} \qquad \eta[u_\theta(a_t) + u_0 \sin\theta] = b\tau_{r\theta},$$

where u_r and u_θ refer to the radial and angular components of the velocity field, and $\tau_{r\theta}$ to the shear stress at the slip plane. For pure slip boundary conditions $b \to \infty$, resulting in [54]:

$$F_s = 6\pi \eta_0 a_t u_0 \left[\frac{2(1 + k_0 a_t) + k_0^2 a_t^2 + k_0^3 a_t^3/3}{3 + k_0 a_t}\right]$$

and

$$F_h = -4\pi \eta_0 a_t u_0 \left[\frac{2\left(1 + k_0 a_t - e^{k_0 a_t}\right) + k_0^2 a_t^2 + k_0^3 a_t^3/3}{k_0^2 a_t^2 (1 + k_0 a_t/3)}\right]. \qquad (12)$$

The friction force experienced by the tracer, equal to $\mathbf{F}_{\rm ext}$, is calculated integrating the stress tensor over the tracer surface. For stick boundary conditions, this leads to [52]:

$$\mathbf{F}_{\rm ext} = 6\pi \eta_0 a_t \mathbf{u_0}\left(1 + k_0 a_t + \frac{1}{9}k_0^2 a_t^2\right). \qquad (13)$$

Note that this expression reduces to Stokes's formula for a Newtonian fluid, $\zeta = 0$ (giving $k_0 = 0$), while in the opposite limit, $k_0 \to \infty$, or $\eta_0 \to 0$, the friction coefficient gives $V_t \zeta_0/2$, with V_t the volume of the tracer.

The velocity profile from the Brinkman equation, Eq. (5), on the other hand, shows a faster decay than the NSE, as shown by the $\sim 1/(k_0^2 r^3)$ dependence at long distances. As expected, for $k_0 = 0$, the $1/r$ decay, typical of a Newtonian fluid within the NSE, is recovered.

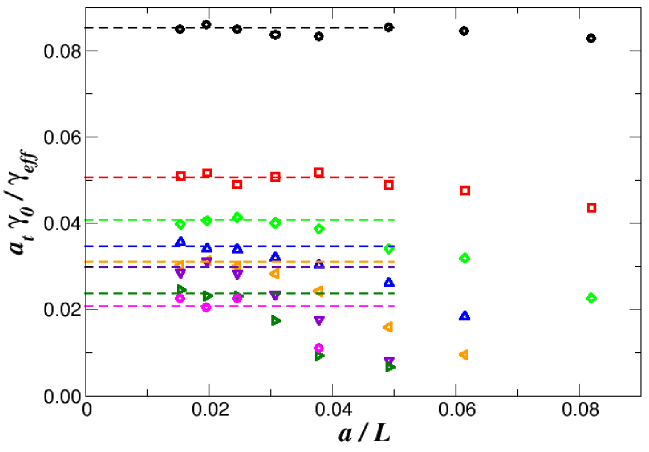


FIG. 2. Inverse friction coefficient as function of the inverse simulation box size for different tracer sizes (different colors and symbols). From top to bottom: $a_t = 1a$, $2a$, $3a$, $4a$, $5a$, $6a$, $7a$, and $8a$.

For slip boundary conditions the friction coefficient is given by:

$$\mathbf{F}_{\rm ext} = 6\pi \eta_0 a_t \mathbf{u_0}\left(\frac{2 + 2k_0 a_t}{3 + k_0 a_t} + \frac{1}{9}k_0^2 a_t^2\right), \qquad (14)$$

which reduces to $4\pi \eta_0 a_t$ for a Newtonian fluid, as expected. In the opposite limit, $k_0 \to \infty$, the boundary condition is not relevant and the friction coefficient is again $V_t \zeta_0/2$.

We end this section by discussing a few important caveats in the connection between the hydrodynamic theory and the Langevin simulations. While one would directly identify ζ_0 with $n\gamma_0$ in Eq. (1), possible differences might be relevant in comparisons. On the one hand, the minimum size of the tracer for the hydrodynamic theory to apply is unknown; and, on the other hand, the compressibility of the colloidal bath (considering only the particles, not the solvent), might be relevant, as the density is diffusive in Langevin systems. Even more, the correct boundary condition on the tracer surface is unknown.

III. RESULTS AND DISCUSSIONS

In this section we first test the theoretical results of the modified NSE with simulations, and then analyze the dynamics of a large forced tracer in a bath of colloidal particles.

A. Friction coefficient of the tracer

The friction coefficient is determined from the steady-state tracer velocity, but due to long-range spatial correlations in the bath, it may show importat finite-size effects. Figure 2 analyzes this effect by showing the inverse effective friction coefficient as a function of the inverse box size for different tracer sizes. This representation is motivated by the theoretical analysis of the finite-size effects in a Newtonian solvent within the NSE. Hasimoto [55] showed that the friction coefficient,

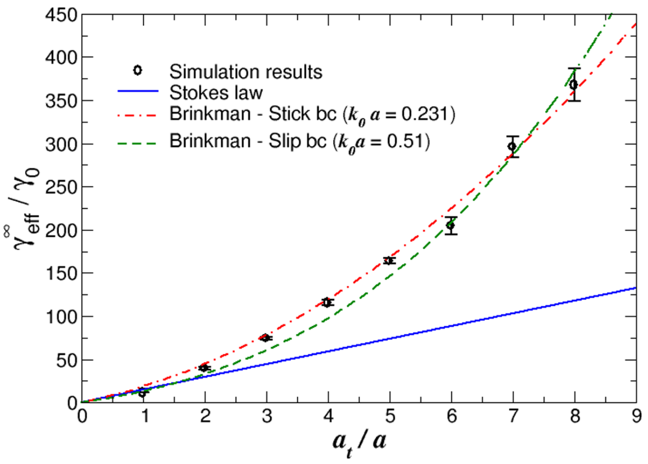


FIG. 3. Friction coefficient extrapolated to the infinite system as function of the tracer size (the error bars indicate the dispersion of the data for large systems). The lines are the results from Brinkman's equation with stick or slip boundary conditions and Stokes's law, as labeled.

$\gamma_{\rm eff}$, experienced by an array of tracers follows:

$$\frac{1}{\gamma_{\rm eff}} = \frac{1}{\gamma_\infty}\left(1 - \frac{\mathcal{C}}{L}\right), \tag{15}$$

where γ_∞ is the friction coefficient measured in an infinite system, $\mathcal{C}$ is a constant that depends on the array structure (simple cubic, BCC, FCC, and so on) and L is the lattice spacing, namely, the simulation box size. For the simple cubic array, that corresponds to the periodic boundary conditions, $\mathcal{C} = 2.8373\, a_t$ [55]. Previous simulations of the diffusion of a tracer in a bath of particles, with microscopic Newtonian dynamics, have shown the validity of this result [56,57]. Furthermore, the value of the friction coefficient extrapolated for the bulk, agrees with the Stokes value using the viscosity (calculated with the Green-Kubo integration of the stress autocorrelation function, as discussed below), and slip boundary conditions.

The data in Fig. 2 shows that $\gamma_{\rm eff}^{-1}$ grows for increasing system sizes for small and intermediate L, but levels off for large systems. These results clearly deviate from the prediction for a Newtonian fluid, Eq. (15), as expected for Langevin systems with a dissipative term. Notably, it also indicates that the bulk value can be obtained from simulations of large enough systems. In a previous work, it was shown that this general result does not depend on the particular details of the simulation [43] (considering the volume of the tracer in the system volume, fixing the center of mass of the system, or varying the friction coefficient with the solvent).

The values of the friction coefficient with an infinite bath, to be compared with the theory, are taken from the plateau for large systems. The results are plotted in Fig. 3 as a function of the tracer size, with the error bars representing the dispersion of the data. The simulation data deviates clearly from the linear trend predicted by Stokes's law for a Newtonian fluid, while the Brinkman equation predicts the qualitative behavior of the friction coefficient adjusting the only unknown parameter k_0 (see below).

To make a more quantitative test of the theoretical models, we calculate the shear viscosity of the bath of quasihard particles. This is given by the Green-Kubo relation, namely the integral of the stress autocorrelation function, which accounts for the particle-particle direct interactions as well as the kinetic energy [48]:

$$\eta_0 = \frac{\beta}{3V}\int_0^\infty dt \sum_{\mu<\nu}\langle \sigma^{\mu\nu}(t)\sigma^{\mu\nu}(0)\rangle, \tag{16}$$

where $\beta = 1/k_B T$ is the inverse thermal energy, V the system volume, and $\sigma^{\mu\nu}(t)$ is the $\mu\nu$ component of the stress tensor. The sum runs over all off-diagonal terms of the stress tensor, and $\langle \sigma^{\mu\nu}(t)\sigma^{\mu\nu}(0)\rangle$ is the stress autocorrelation function. The time integral over the correlation function is more conveniently performed using the Einstein relation [58,59].

The Green-Kubo integration gives for the viscosity of the bath $\eta_0 = (3.9 \pm 0.1)\sqrt{kTm/a^2}$. With this value, the Stokes prediction is plotted in Fig. 3 (blue continuous line), which underestimates notably the simulation data for large tracers, although the small size limit is correctly captured. The friction coefficient obtained from the Brinkman equation has been adjusted to reproduce the simulations, using k_0 as fitting parameter. The dashed lines in Fig. 3 show the fittings with the calculations considering stick or slip boundary conditions (red or green lines, respectively). Both fittings are equally acceptable, but they give different values of the fitting parameter k_0, as shown in the figure.

From the simulation, identifying $\zeta_0 = n\gamma_0$, we expect $k_0 = \sqrt{n\gamma_0/\eta_0} = 0.39/a$, which is within the range of values provided by both fittings. A small value of k_0 corresponds to a system controlled by the viscosity of the bath of particles, as expected due to the high density of the bath (recall that the volume fraction is $\phi = 0.50$).

To further compare the model and the simulations, we study the velocity profile in the bath. Figure 4 shows the velocity of the bath particles in front of the tracer for two tracer sizes and the system with $N = 15\,625$ particles (only the radial component is studied). The distribution of bath particles surrounding the tracer, $\rho(r)$, is also included in the figure to facilitate the interpretation. The velocity profile oscillates in phase with the bath density, and decays faster than the inverse distance, the prediction for the Newtonian fluid, irrespective of the boundary condition. Brinkman's model, Eq. (5), on the other hand, reproduces quite well the decay of the velocity profile (as $1/r^3$), but also quantitatively with the values of k_0 obtained from the fitting of the friction coefficient for both boundary conditions, and for both tracer sizes. However, the theory fails to capture the oscillations due to the finite size of the bath particles, as expected for a continuum model for the bath. Again, both boundary conditions compare equally well with the simulations, bracketing the simulation results.

A more prominent difference between the stick and slip boundary conditions is obtained if the angular component of the velocity field in the direction perpendicular to the external force is studied. This is tackled in Fig. 5 for the same tracer sizes (the z component of the velocity, parallel to the force, is studied). For small distances from the tracer, the stick boundary conditions result in a positive velocity,

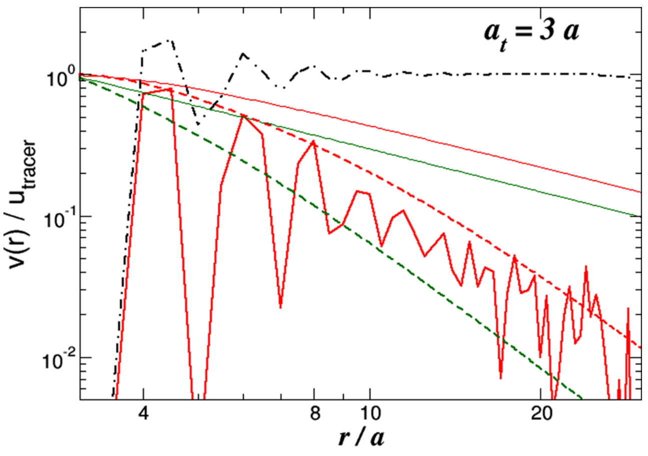


FIG. 4. Velocity profile in the colloidal bath in front of the tracer from simulations (continuous red line), for two tracer radii, as labeled. Theory results for a Newtonian fluid (thin red and green lines) and the Brinkman equation with stick or slip boundary conditions (dashed red and green lines, respectively) are also included. The dash-dotted black line represents the density of bath particles around the tracer.

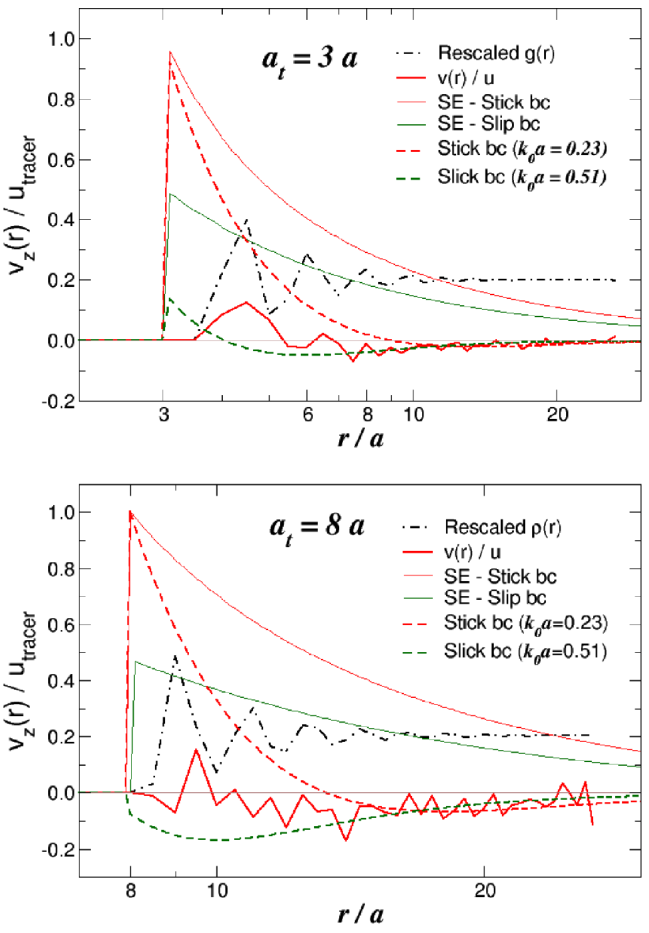


FIG. 5. The z component of the velocity in the colloidal bath in the plane perpendicular to the tracer for $a_t = 3a$ (upper panel) and $a_t = 8a$ (lower panel). Simulations (continuous red line), and theory results for a Newtonian fluid (thin red and green lines) and the Brinkman equation with stick or slip boundary conditions (dashed red and green lines, respectively) are shown. The dash-dotted black line represents the density of bath particles around the tracer rescaled to fit into the same scale.

which becomes negative further away, but the slip boundary condition produces a negative velocity for all distances. The simulation results agree with both cases for long distances (negative velocity), but are close to zero near the tracer. This result, in conjuction with all previous comparisons, probably indicates that a mixed boundary condition is optimal in describing the friction and velocity fields of the tracer in a colloidal bath with the Brinkman equation. For completeness, the predictions from the NSE for stick and slip boundary conditions are shown, indicating that the behavior observed in the simulations cannot be reproduced.

B. Tracer dynamics

In this subsection, we analyze the transient dynamics of the forced tracers of different sizes, for a small pulling force. Figure 6 shows the mean-squared displacement of the tracer perpendicular to the force direction and parallel to it (with the drift velocity substracted). Long-time diffusion is reached for all tracers, in particular in the longitudinal direction, i.e., superdiffusion is not observed for this density [16] (superdiffusion has been indeed observed in this same system for larger densities). Notably, the self diffusion coefficient decreases with increasing tracer size, developing a shoulder in the MSD and a sublinear increase at intermediate times. The typical distance corresponding to the height of the shoulder also decreases with the size of the tracer. Recall that the length unit is the bath particle radius, i.e., if the tracer radius is used, the decrease in the localization length is enlarged, pushing to a tiny fraction of the tracer radius (smaller than $10^{-4}a_t^2$ for the biggest tracer).

The self-diffusion coefficients, obtained from the long-time slope of the MSD in both directions, are shown in Fig. 7. Both of them are very similar and follow the same trend, decaying almost two decades in the range of tracer sizes studied here. Indeed, not only the slopes of the MSD in both directions are close to each other, but the MSD themselves are very similar (the relative differences are below 20% in all cases, and constant within the statistical noise). The equality of the MSD

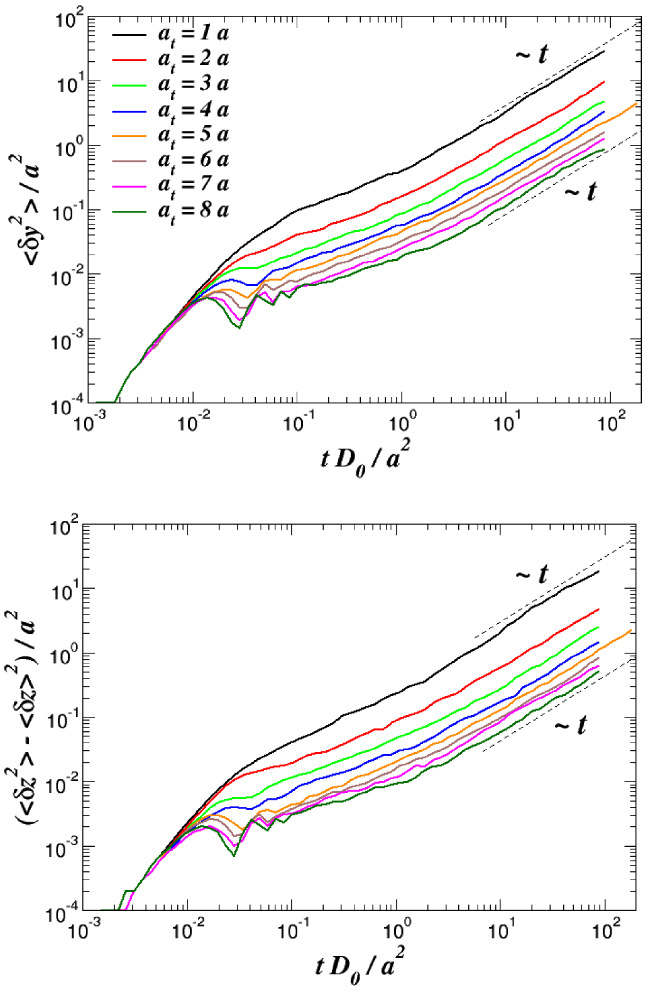


FIG. 6. Tracer mean-squared displacement in the direction perpendicular to the force (upper panel), and parallel to the force (lower panel), for different tracer radii, as labeled (increasing from top to bottom).

in both directions, and the concomitant diffusion coefficients, despite the anisotropy induced by the external force, indicates

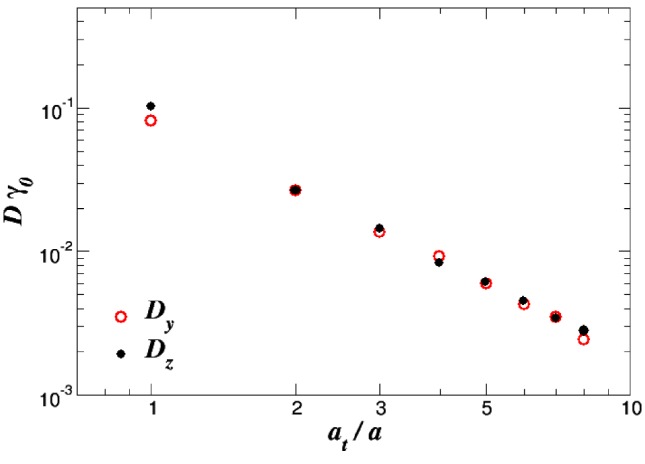


FIG. 7. Diffusion coefficients in the direction perpendicular and parallel to the external force, as labeled.

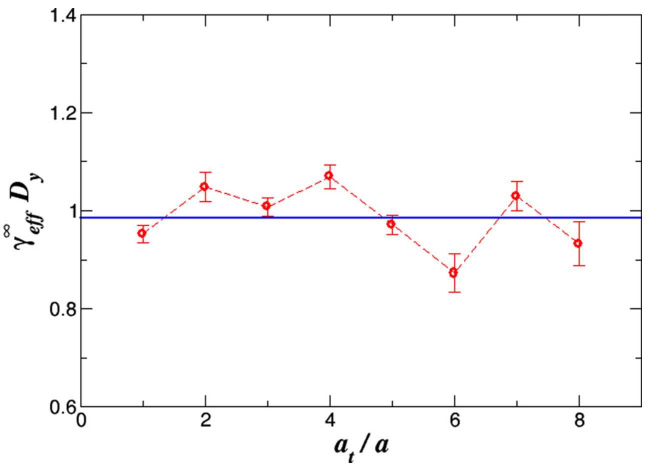


FIG. 8. Diffusion coefficient in the force direction times the friction coefficient. The blue line is the average over all data.

that the force is small enough to keep the system in the linear regime.

Finally, we check the Stokes-Einstein relation for the tracer by plotting the product of the diffusion coefficient times the friction coefficient for all tracer radii. Figure 8 shows these results as a function of the tracer size. The product is close to 1 in all cases, fluctuating around a mean value of 0.986, confirming the validity of the Stokes-Einstein relation, or stated more generally, of the linear response formalism. The mobility, viz. the inverse friction coefficient, of a tracer feeling a small force is proportional to the diffusion coefficient of the unforced tracer, and the prefactor is given by the thermal energy, which is set to unity in the simulations.

IV. CONCLUSIONS

The dynamics of a large tracer pulled with a small force in a bath of quasihard colloidal spheres has been studied with Langevin dynamics simulations, and with continuum mechanics. The force is small enough to keep this out-of-equilibrium system in the linear response regime. The analysis of finite-size effects in the simulations has shown that the correlations in the bath, induced by the moving tracer, decay faster than in a Newtonian fluid, and within the simulation box, if the system is large enough. This has allowed the analysis of the microviscosity without futher extrapolation with the theory. The Navier-Stokes equation has been modified, adding a term proportional to the fluid velocity, resulting in an equation identical to the Brinkman equation, albeit our interpretation of the terms is different. This two-fluid model provides a length scale, k_0^{-1}, for the crossover from diffusive transverse momentum transport to friction with the solvent, which depends on the viscosities of the two fluids. The resulting friction coefficient for the tracer grows faster than linear, with both stick and slip boundary conditions, and the velocity profile decays as $\sim 1/r^3$ for finite k_0. The results for a Newtonian fluid are recovered in the limit $k_0 \to 0$.

The comparison of the simulations and theory gives semi-quantitative agreement. Fitting k_0, the simulation data can be reproduced with the model, both the friction coefficient and velocity profile in the bath for long distances. The value of k_0 also corresponds to the expectation based on the viscosity calculated from the Green-Kubo relation and the solvent friction coefficient. The two-fluid model describes satisfactorily the physical phenomena in colloidal microrheology and shows that a correct interpretation of the results requires accounting for colloidal bath particles and solvent. Also, our results apparently point to mixed effective boundary conditions between stick and slip.

The fluctuations of the tracer position have been studied to obtain the mean-squared displacement in the direction parallel to the force and perpendicular to it. Diffusion is attained in both cases at long times, after a transient trapping with a typical length decreasing for increasing tracer sizes. Because the system is in the linear response regime, the diffusion coefficients in both directions are similar despite the anisotry provoked by the external force. Furthermore, the Stokes-Einstein relation is fulfilled, confirming the validity of linear response.

ACKNOWLEDGMENTS

We are indebted to Professor Thomas Franosch for many useful discussions on the Brinkman model and its solutions. A.M.P. acknowledges financial support from the Spanish Ministerio de Ciencia and FEDER under Project No. PGC2018-101555-B-I00 and from UAL/CECEU/FEDER (UAL18-FQM-B038-A); F.O., G.O., and E.M.G. also appreciate the support from Project No. RTI2018-095993-B-I00. F.O. is supported by an FPI Fellowship (attached to Project No. TIN2015-66680-C2-1-R) from the Spanish Ministerio de Ciencia.

[1] P. M. Chaikin and T. C. Lubensky, *Principles of Condensed Matter Physics* (Cambridge University Press, Cambridge, UK), 1995.
[2] J. K. G. Dhont, *An Introduction to Dynamics of Colloids* (Elsevier, Amsterdam, 1996).
[3] Ed. A. Fernandez-Nieves, A. M. Puertas, *Fluids, Colloids and Soft Materials* (John Wiley & Sons, New York, 2016).
[4] T. G. Mason, D. A. Weitz, Phys. Rev. Lett. **74**, 1250 (1995).
[5] P. Cicuta and A. M. Donald, Soft Matter **3**, 1449 (2007).
[6] L. Wilson and W. C. K. Poon, Phys. Chem. Chem. Phys. **13**, 10617 (2011).
[7] A. M. Puertas and Th. Voigtmann, J. Phys.: Condens. Matter **26**, 243101 (2014).
[8] E. Furst and T. Squires, *Microrheology* (Oxford University Press, Oxford, UK, 2017).
[9] P. Habdas, D. Schaar, A. C. Levitt, and E. R. Weeks, Europhys. Lett. **67**, 477 (2004).
[10] I. Sriram, E. M. Furst, R. J. DePuit, and T. M. Squires, J. Rheol. **53**, 357 (2009).
[11] I. Sriram, A. Meyer, and E. M. Furst, Phys. Fluids **22**, 062003 (2010).
[12] I. C. Carpen and J. F. Brady, J. Rheol. **49**, 1483 (2005).
[13] A. S. Khair and J. F. Brady, J. Fluid Mech. **557**, 73 (2006).
[14] I. Gazuz, A. M. Puertas, Th. Voigtmann, and M. Fuchs, Phys. Rev. Lett. **102**, 248302 (2009).
[15] R. N. Zia and J. F. Brady, J. Rheol. **56**, 1175 (2012).
[16] D. Winter, J. Horbach, P. Virnau, and K. Binder, Phys. Rev. Lett. **108**, 028303 (2012).
[17] R. N. Zia, Annu. Rev. Fluid Mech. **50**, 371 (2018).
[18] R. G. Larson, *The Structure and Rheology of Complex Fluids* (Oxford University Press, New York, 1999).
[19] J. F. Brady, J. Chem. Phys. **99**, 567 (1993).
[20] P. Strating, Phys. Rev. E **59**, 2175 (1999).
[21] T. M. Squires and J. F. Brady, Phys. Fluids **17**, 073101 (2005).
[22] R. J. DePuit, A. S. Khair, and T. M. Squires, Phys. Fluids **23**, 063102 (2011).
[23] S Leitmann, T Franosch, Phys. Rev. Lett. **118**, 018001 (2017).
[24] S. Leitmann and Th. Franosch, Phys. Rev. Lett. **111**, 190603 (2013).
[25] S. Leitmann, S. Mandal, M. Fuchs, A. M. Puertas, and Th. Franosch, Phys. Rev. Fluids **3**, 103301 (2018).
[26] T. M. Squires, Langmuir **24**, 1147 (2008).
[27] A. J. Levine and T. C. Lubensky, Phys. Rev. Lett. **85**, 1774 (2000).
[28] A. J. Levine and T. C. Lubensky, Phys. Rev. E **63**, 041510 (2001).
[29] B. U. Felderhof, J. Chem. Phys. **131**, 164904 (2009).
[30] H. C. W. Chu and R. N. Zia, J. Colloid Interface Sci. **539**, 388 (2019).
[31] I. Gazuz and M. Fuchs, Phys. Rev. E **87**, 032304 (2013).
[32] M. Gruber, G. C. Abade, A. M. Puertas, and M. Fuchs, Phys. Rev. E **94**, 042602 (2016).
[33] M. Gruber, A. M. Puertas, and M. Fuchs, Phys. Rev. E **101**, 012612 (2020).
[34] F. Vogel, A. Zippelius, and M. Fuchs, Europhys. Lett. **125**, 68003 (2019).
[35] H. C. Brinkman, Appl. Sci. Res. **1**, 27 (1947).
[36] C. F. E. Schroer and A. Heuer, Phys. Rev. Lett. **110**, 067801 (2013).
[37] G. Gradenigo, E. Bertin, and G. Biroli, Phys. Rev. E **93**, 060105(R) (2016).
[38] T. Wang and M. Sperl, Phys. Rev. E **93**, 022606 (2016).
[39] R. Wulfert, A. U. Seifert, and T. Speck, Soft Matter **13**, 9093 (2017).
[40] S. Marenne, J. F. Morris, D. R. Foss, and J. F. Brady, J. Rheol. **61**, 477 (2017).
[41] E. Lange, J. B. Caballero, A. M. Puertas, and M. Fuchs, J. Chem. Phys. **130**, 174903 (2009).
[42] W. Paul and D. Y. Yoon, Phys. Rev. E **52**, 2076 (1995).
[43] F. Orts, G. Ortega, E. M. Garzón, and A. M. Puertas, Comput. Phys. Commun. **236**, 8 (2019).
[44] G. Ortega, A. M. Puertas, and E. M. Garzón, J. Supercomput. **73**, 370 (2017).
[45] G. Ortega, A. M. Puertas, F. J. de Las Nieves, and E. M. Garzón, *GPU Computing to Speed-Up the Resolution of Microrheology Models*, International Conference on Algorithms and Architectures for Parallel Processing, Lecture Notes in Computer Science Vol. 10048 (Springer, Cham, 2016), pp. 457-466.

[46] I. V. Morozov, A. M. Kazennov, R. G. Bystryi, G. E. Norman, V. V. Pisarev, and V. V. Stegailov, Comput. Phys. Commun. **182**, 1974 (2011).

[47] V. Sels, J. Coelho, A. M. Dias, and M. Vanhoucke, Comput. Operat. Res. **53**, 107 (2015).

[48] D. J. Evans, G. P. Morriss, *Statistical Mechanics of Nonequilibrium Liquids* (Australian National University E-Press, Camberra, Australia, 2007).

[49] W. Hess and R. Klein, Adv. Phys. **32**, 173 (1983).

[50] C. K. W. Tam, J. Fluid Mech. **38**, 537 (1969).

[51] L. Durlofsky and J. F. Brady, Phys. Fluids **30**, 3329 (1987).

[52] C. Pozrikidis, *Introduction to Theoretical and Computational Fluid Dynamics* (Oxford University Press, Oxford, 2011).

[53] E. Guyon, J. P. Hulin, L. Petit, and C. D. Mitescu, *Physical Hydrodynamics* (Oxford University Press, Oxford, UK, 2001).

[54] B. U. Felderhof, Phys. Fluids **19**, 126101 (2007).

[55] H. Hasimoto, J. Fluid Mech. **5**, 317 (1959).

[56] I.-C. Yeh and G. Hummer, J. Phys. Chem. B **108**, 15873 (2004).

[57] R. O. Sokolovskii, M. Thachuk, G N. Patey, J. Chem. Phys. **125**, 204502 (2006).

[58] A. M. Puertas, C. de Michele, F. Sciortino, P. Tartaglia, and E. Zaccarelli, J. Chem. Phys. **127**, 144906 (2007).

[59] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids* (Clarendon, Bristol, UK, 1987).

2.2.3 On solving the unrelated parallel machine scheduling problem: active microrheology as a case study

Title	On solving the unrelated parallel machine scheduling problem: active microrheology as a case study		
Authors	*F. Orts, G. Ortega, A.M. Puertas, I. García and E.M. Garzón*		
Journal	Journal of Supercomputing		
Year	2020		
Volume	76		
Pages	8494-8509		
DOI	https://doi.org/10.1007/s11227-019-03121-z		
IF (JCR 2019)	2.474		
Categories	Computer Science, Theory & Methods:	33/110	**(Q2)**
	Computer Science, Hardware & Architecture:	26/53	(Q2)
	Engineering, Electrical & Electronic:	139/273	(Q3)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

The Journal of Supercomputing
https://doi.org/10.1007/s11227-019-03121-z

On solving the unrelated parallel machine scheduling problem: active microrheology as a case study

F. Orts[1] · G. Ortega[1] · A. M. Puertas[2] · I. García[3] · E. M. Garzón[1]

Abstract
Modern computational platforms are characterized by the heterogeneity of their processing elements. Additionally, there are many algorithms which can be structured as a set of procedures or tasks with different computational cost. Balancing the computational load among the available processing elements is one of the main keys for the optimal exploitation of such heterogeneous platforms. When the processing time of any procedure executed on any of the available processing elements is known, this workload-balancing problem can be modeled as the well-known *scheduling on unrelated parallel machines* problem. Solving this type of problems is a big challenge due to the high heterogeneity on both, the tasks and the machines. In this paper, the balancing problem has been formally defined as a global optimization problem which minimizes the makespan (parallel runtime) and a heuristic based on a Genetic Algorithm, called Genetic Scheduler (GenS), has been developed to solve it. In order to analyze the behavior of GenS for several heterogeneous clusters, an example taken from the field of statistical mechanics has been considered as a case study: an active microrheology model. Given this type of problem and a heterogeneous cluster, we seek to minimize the total runtime to extend and analyze in depth the case of study. In such context, a task consists of the simulation of a tracer particle pulled into a cubic box with smaller bath particles. The computational load depends on the total number of the bath particles. Moreover, GenS has been compared to other dynamic and static scheduling approaches. The experimental results of such a comparison show that GenS outperforms the rest of the tested alternatives achieving a better distribution of the computational workload on a heterogeneous cluster. So, the scheduling strategy developed in this paper is of potential interest for any application which requires the execution of many tasks of different duration (a priori known) on a heterogeneous cluster.

Keywords Parallel scheduling · Heterogeneous cluster · Unrelated machines · Genetic Algorithm

✉ G. Ortega
gloriaortega@ual.es

Extended author information available on the last page of the article

Published online: 02 January 2020

1 Introduction

Modern computational systems consist of heterogeneous clusters which are composed by the interconnection of Processing Units (PUs) with different computational power, such as CPU-cores, GPUs and so on [1]. Algorithms developed for this kind of platforms have to treat such heterogeneity to efficiently exploit the different resources on modern computers. To this effect, the programmer is responsible for explicitly selecting the devices and mapping the tasks among PUs. So, scheduling techniques become one of the most challenging problems, having a tremendous impact on performance. Many examples of parallel applications consist of a set of independent tasks, with different computational cost, which have to be scheduled on a set of heterogeneous processing units in an optimal way. This problem can be modeled as a *scheduling tasks on unrelated parallel machines* problem, which is NP-complete [2] and very well known in the field of operational research [3].

There are two different approaches for the scheduling problems, dynamic and static. The dynamic one is based on the definition of a global queue of tasks from which every available PU picks a new task up. A dynamic scheduling does not need any a priori information and usually is the best option when the tasks load is unpredictable. However, dynamic scheduling can produce non-optimal solutions when the tasks runtime is strongly heterogeneous. Several dynamic interfaces have emerged in the last few years to face scheduling in heterogeneous clusters. For instance, StarPU [4], Qilin [5] and Scout [6] offer different methods to map tasks to CPU and GPU. The disadvantage of these paradigms is that they require the programmers to rewrite their codes using a new programming language in the case of StarPU or Scout or using specific APIs in Qilin [7].

On the other hand, static approaches are useful when it is possible to have an estimation of the runtime of the tasks a priori. In such cases, they can provide results near optima since they consider the problem from a holistic view. This paper is focussed on such context. Although this kind of problems is challenging, it can be efficiently solved if an a priori knowledge about the runtime of every task on every machine is considered. The proposed scheduling is of potential interest for any problem that meets the above-mentioned premises.

In this work, an active microrheology model (AMM) in hard colloids has been selected as a case study to illustrate the scheduling of simulations of bulk systems on heterogeneous platforms. From the computational point of view, simulations of bulk systems have huge requirements and can be structured as a set of independent tasks with different computational loads (simulations of systems with different sizes).

Here, a finite size analysis is used to extrapolate the results from a finite system to an infinite one, requiring simulations of systems with different (large) sizes. In active microrheology, the mechanical and flow behavior of a complex fluid is studied at the microscopic level [8, 9]. Therefore, in order to compute the microviscosity for a bulk system, it is necessary to run simulations of systems with different sizes, and extrapolate to the infinite system relying on the model. Note that for every system size, many tracer trajectories must be evaluated (typically 500 in this work) to obtain a good estimation of the average tracer velocity. In the context of AMM simulations, it is feasible to have a good a priori estimation of the simulation time on different processing units.

So, a static scheduling based on a global analysis is an appropriate option to optimize the parallel execution of such simulations. In this paper, the scheduling strategy is formally defined as a global optimization problem which minimizes the makespan (parallel runtime of the simulation processes). Then, a heuristic based on a Genetic Algorithm is developed to solve the scheduling on unrelated parallel machines. Hereinafter, it is referred to as the Genetic Scheduler (GenS). Other scheduling approaches (two dynamic and two static strategies) are revised and comparatively evaluated with respect to GenS. Our results show that GenS outperforms the other scheduling methods in terms of makespan using the paradigmatic case study.

The main contributions of the paper can be summarized as follows: (1) a new scheduling heuristic based on a Genetic Algorithm to efficiently distribute the heterogeneous tasks on heterogeneous resources has been designed and comparatively evaluated; (2) this scheduling makes feasible to solve computationally harder simulations of the active microrheology case study; (3) a scheduling software to efficiently distribute a set of independent tasks with different costs on heterogeneous processing units, called GenS, is provided (https://github.com/2forts/GENS). Thus, this software can be useful for all problems which can be modeled by scheduling on unrelated parallel machines beyond the case study of this paper.

2 Scheduling problem on unrelated parallel machines

Let assume that a cluster has K PUs (Processing Units), that is, for example the total number of available CPU-cores plus GPUs. Let $\{R_m\}$ be the set of tasks that defines the model, with $m = 1, \dots, M$ and $M = \sum_{i=1}^{I} Q_i$ represents the total number of tasks to compute, I denotes the number of different system sizes N_i, with $1 \leq i \leq I$, and Q_i represents the number of tasks with system size N_i. Then, the goal is to find a scheduling that minimizes the makespan, C_{max}

$$\begin{aligned} \text{Find:}\quad & X \\ \text{to minimize:}\quad & C_{max} \\ \text{subject to:}\quad & t_k = \sum_{i=1}^{I} x_{k,i} t_{k,i} \leq C_{max}, 1 \leq k \leq K \\ & \sum_{k=1}^{K} x_{k,i} = Q_i, 1 \leq i \leq I \\ & x_{k,i} \in \{0, 1, \dots, Q_i\}, 1 \leq k \leq K; 1 \leq i \leq I \end{aligned} \tag{1}$$

where $x_{k,i}$ represents the number of tasks of size N_i assigned to the k-th PU; $x_{k,i}$ is an element of the matrix, X, that defines the assignment of tasks to PUs (machines); $t_{k,i}$ represents the runtime to compute a task of system size N_i on the k-th PU. The constraints for X mean that every task is computed on a single PU and every set of Q_i tasks with the same size is distributed among all the PUs. The k-th row of X defines the set of tasks assigned to k-th PU, and the i-th column establishes the distribution of the tasks of size N_i among the K PUs (see Fig. 1).

The scheduling problem defined by Eq. 1 includes the runtime of every task at every PU, $t_{k,i}$. The estimation of $t_{k,i}$ can be accurately and fast computed a priori, because the $K \times I$ matrix $T = (t_{k,i})$ includes a high percentage of identical rows related to the same kinds of PUs. The runtime of the tasks can be characterized by a matrix $\mathcal{T}$ of reduced dimensions $S \times I$ where S represents the number of different kinds of PUs.

The scheduling on unrelated parallel machines is a challenge because of the heterogeneity of both, the required tasks and cluster architecture. Then, it is necessary to define an appropriate task scheduling to obtain the optimal parallel performance.

3 Scheduling approaches

According to the formalism introduced, the static methodology to optimize the task distribution among the heterogeneous PUs consists of three stages: (1) profiling stage, which estimates the values of every element of the matrix T, according to the different sizes of the systems involved in the analysis and the number of Processing Units, PUs; (2) optimal scheduling estimation to identify the set of tasks which every PU should compute, bearing in mind the a priori knowledge provided by the profiling stage, so a parallel runtime can also be estimated; and (3) parallel execution of all simulations on the heterogeneous PUs of the cluster according to the scheduling defined in the second stage.

The optimal scheduling estimation (stage 2) could be simpler if there was a single type of tasks and PUs, since we could easily reach a good solution using a homogeneous distribution. However, in general, this estimation is more complex, even with

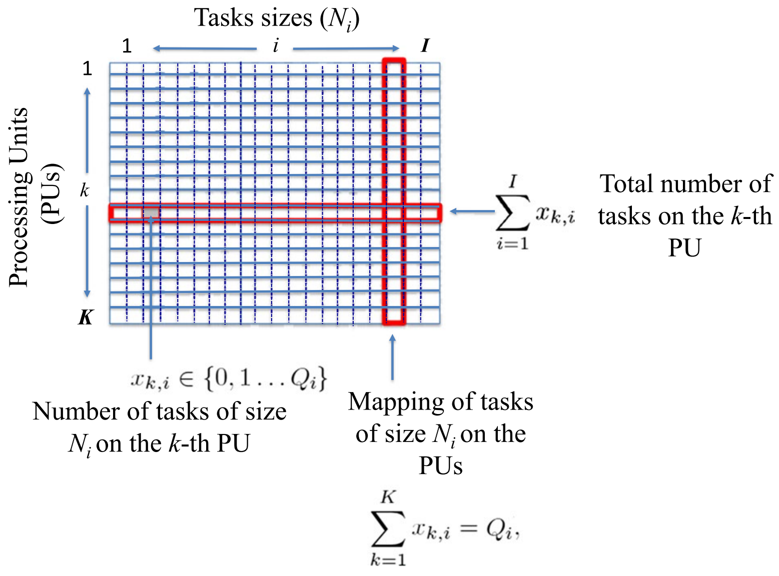

Tasks sizes (N_i)
1
i
I
1
k
K
Processing Units (PUs)
$\sum_{i=1}^{I} x_{k,i}$
Total number of tasks on the k-th PU
$x_{k,i} \in \{0, 1 \ldots Q_i\}$
Number of tasks of size N_i on the k-th PU
Mapping of tasks of size N_i on the PUs
$\sum_{k=1}^{K} x_{k,i} = Q_i,$

Fig. 1 Matrix X defines the assignment of tasks to PUs

a single type of PU, because the parallel software may include tasks with different computational loads. Of course, the computational complexity of the optimal scheduling estimation increases for high values of I, M and K. The scheduling on a heterogeneous cluster is NP-complete [10]. Our goal is to apply a heuristic which provides near-optimal solutions for the scheduling problem [10].

3.1 Genetic scheduler (GenS)

Genetic Algorithms (GAs) have been widely used for the resolution of scheduling problems. For instance, GAs are used to solve scheduling problems according to a static scheme in Sels et al. [11]. A scheduling algorithm based on double-fitness adaptive algorithm-job spanning time and load-balancing genetic algorithm is applied in cloud computing in Wang et al. [12]. Another GA for the same case was presented by Sharma et al. [13]. Adan et al. present a GA for production scheduling at a back-end production of electronics manufacturing [14]. A GA to schedule non-preemptive tasks onto identical multiprocessors was presented in Al-Said et al. [15]. Cappadona et al. afford the unrelated parallel machine scheduling with limited and differently skilled human resources [16]. A stochastic search method based on a GA approach was presented in Hou et al. [17]. Jooyayeshendi et al. and Page et al. presented GAs for solving the unrelated parallel machine scheduling on heterogeneous distributed systems, in a dynamic context [18, 19]. The proposal of Kaiser et al. affords the scheduling on homogeneous clusters [20]. A GA for the unrelated parallel machine scheduling problem but with sequence-dependent setup times was presented in Vallada et al. [21].

In this work, a GA is customized for solving the unrelated parallel machine scheduling on heterogeneous clusters for the specific case defined by Eq. 1. A GA works with a set of individuals which represents possible solutions of the scheduling policy problem (*population*). It is an iterative procedure which starts with a random set of individuals, $P(0)$, and at every iteration, *iter*, a selection mechanism and genetic operators are applied to the population, $P(iter)$. Thus, the population is constantly evolving. The selection mechanism allows the individuals of the next generation to be closer to the optimal solution (see Algorithm 1).

Algorithm 1 Genetic Algorithm

```
1: $iter \leftarrow 0$
2: Initialize random population $P(0)$
3: Evaluate the fitness for the population $P(0)$
4: while termination condition is not true do
5:     $iter \leftarrow iter + 1$
6:     Select $P(iter)$ from $P(iter - 1)$
7:     Apply genetic operators (crossover and mutation) to $P(iter)$
8:     Evaluate the fitness for $P(iter)$
9: end
```

To apply Algorithm 1 to the problem of finding a near-optimal scheduling, it is necessary to specify the following concepts: individual, fitness function, and genetic operators (crossover and mutation). We propose to adapt a GA to the aforementioned scheduling problem, resulting in the GenS algorithm.

Bearing in mind the formalism introduced above, every individual in $P(iter)$ is represented by a $K \times I$ matrix X which defines the assignment of tasks to the PUs according to the definition in Eq. 1 (and Fig. 1). After the evaluation of the fitness function (UB) for the whole population, individuals are ordered according to their fitness. Thus, the individuals with smaller UB will be selected while the GA advances.

At every iteration of Algorithm 1, two operations are applied to the population to promote the evolution. Firstly, a random set of pairs of individuals (parents) is defined and then, new individuals (children) are produced by the crossover operator. Then, the well-known single point crossover operator is applied. Figure 2 describes how the crossover is applied. A random column is selected to split the matrices of both parents, and new individuals are generated swapping the four sets of columns. In this scheme, the children can be considered as valid solutions since the constraints for the columns of their matrices are verified. After the crossover, the mutation operator acts on every descendant and it can alter the distribution of every column (with a probability of 1%). It is a random exchange of tasks of the same size between a pair of PUs, i.e., elements in the same column of the corresponding matrix interchange their tasks partially. Every iteration starts with the same population size (PS). The selection phase only consists of choosing/ keeping the best PS individuals since the population has been previously ordered according to the fitness, UB. The procedure stops when the UB over 10 iterations does not change for the 30% of best individuals. Summarizing, if $\{t_{k,i}\}$ with $1 \leq k \leq K$ and $1 \leq i \leq I$ is known, GenS is able to identify a near-optimal distribution of tasks among the set of PUs.

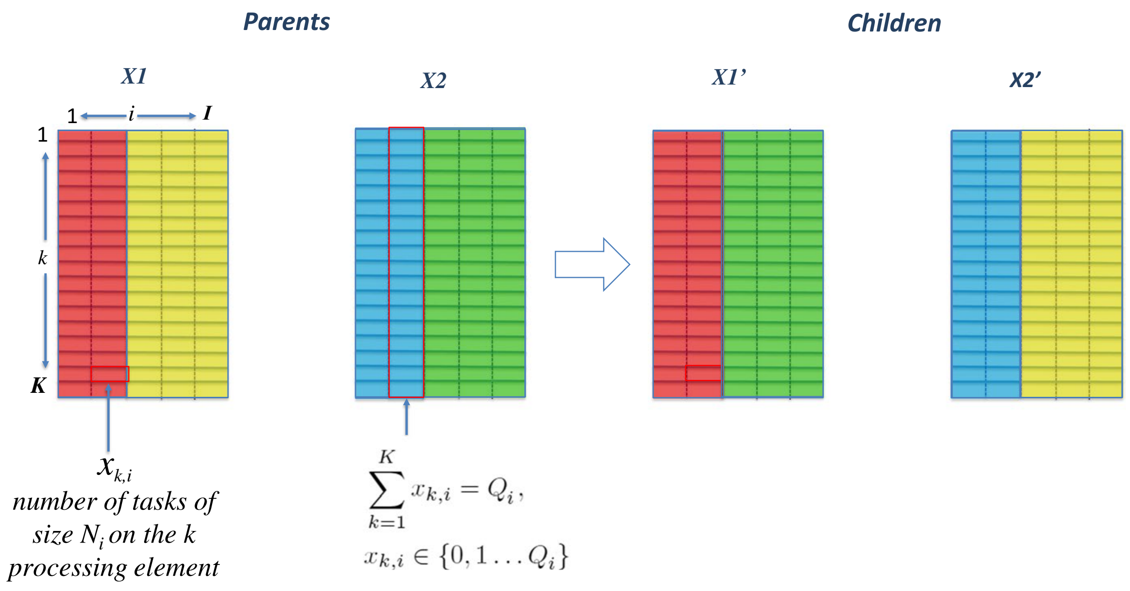

Parents
Children
X1
X2
X1'
X2'
1
i
I
1
k
K
$X_{k,i}$
number of tasks of size N_i on the k processing element
$\sum_{k=1}^{K} x_{k,i} = Q_i,$
$x_{k,i} \in \{0, 1 \ldots Q_i\}$

Fig. 2 Crossover procedure to produce new individuals in the population

3.2 Additional static approaches

An example of static approach is the Polynomial Time Approximations Scheme (PTAS) algorithm [10]. The PTAS algorithm can give a good estimation of the optimal scheduling, with the additional advantage that it is possible to estimate theoretically the ratio to the optimal solution. However, PTAS has a high computational overhead due to the large amount of information that it is necessary to store. This is the reason PTAS has not been included in the comparative study of GenS.

An alternative approach to GenS consists of a cyclic distribution of the tasks over the set of PUs. First of all, the tasks are ordered according to their computational load. After that, they are distributed in a cyclic order among the PUs. In this way, every PU computes similar percentages of tasks of different costs. Hereinafter, this scheme will be referred to as Cyclic. It is probable that it achieves a near-optimal schedule in homogeneous clusters.

A greedy heuristic following the scheme considered in [22] can also be defined to solve the scheduling problem (Greedy). For a given system size, (N_i), the a priori estimation of the runtime allows us to identify the slowest PU, and the acceleration factor of the remaining PUs with respect to it. These factors define the percentage of tasks of a specific size that will be executed in every PU. So, every set of Q_i tasks with the same workload is distributed among the PUs. The PUs with less computational power compute fewer simulations, and PUs with more power will compute the percentage of tasks defined by the corresponding acceleration factor. This procedure is repeated for every subset of tasks, obtaining a near-optimal distribution in every case, with the aim of obtaining a global optimal solution.

3.3 Dynamic approaches

Several kinds of scheduling policies without a priori estimations of the tasks runtime can be defined. However, our interest is focussed on two dynamic approaches that partially use this information, since the starting point of both is an ordered tasks queue according to their computational load, N_i.

The Simple Tasks Queue (STQ) solves the problem dynamically, managing a single queue. Each PU will compute a task from this queue, and when it finishes, it takes the new task from the head of the queue.

Another dynamic approach is to use a Double-Ended Queue in combination with a classification of the devices in two categories, slow and fast devices. In this scheme, slower devices take the lighter tasks of the queue, and faster devices the heavier ones. It will be referred to as Double-Ended Tasks Queue (DETQ), and it considers a priori information about the loads of tasks and the power of machines.

4 Active microrheology model (AMM) as a case study

As mentioned above, AMM is considered here as case study because from the computational of view it can be seen as a set of independent tasks of several different loads which can be executed on heterogeneous clusters. In active microrheology, the mechanical and flow behavior of a complex fluid is studied at the microscopic level [9, 23]. For this, an intruder particle, typically of colloidal size, is introduced and pulled through the system, and its dynamics is monitored. In particular, the microviscosity can be computed from the stationary tracer velocity at long times.

In our case study, the host fluid is modeled as Brownian quasi-hard spheres, mimicking hard colloids. Brownian motion is described by the Langevin equation [24], which for particle j reads:

$$m_j \frac{d^2\mathbf{r}_j}{dt^2} = \sum_i \mathbf{F}_{ij} - \gamma_j \frac{d\mathbf{r}_j dt}{+} \boldsymbol{\eta}_j(t) \quad + \mathbf{F}_{\mathrm{ext}}\delta_{j1} \tag{2}$$

where the terms in the r.h.s. are the interaction forces ($\sum_i \mathbf{F}_{ij}$), friction with the solvent ($-\gamma_j \frac{d\mathbf{r}_j}{dt}$), random force ($\boldsymbol{\eta}_j(t)$), and external force ($\mathbf{F}_{\mathrm{ext}}\delta_{j1}$), respectively; γ_j is the friction coefficient with the solvent, which is related to the random force via the fluctuation dissipation theorem [24], and depends linearly on the particle radius. The external force, $\mathbf{F}_{\mathrm{ext}}$, which acts only onto the tracer, labeled by $j = 1$, is constant in our model (this fact is expressed by the well-known Kronecker delta, denoted by δ_{j1}). The interaction forces are derived from the interparticle potential $V(r_{ij}) = k_B T(r_{ij}/d_{ij})^{-36}$, where r_{ij} is the center to center distance, and $d_{ij} = (a_i + a_j)/2$, where a_i is the radius of particle i.

The simulations are run in a cubic box, with N particles and periodic boundary conditions. The bath particles and tracer have radii a_b and a_t, respectively, and all particles have the same mass, m. Details of the features of the simulations can be found in [25]. Figure 3 presents a snapshot of a system with $N = 15,625$ particles.

In the simulations, the tracer particle is pulled at a constant force, and its trajectory is recorded. The effective friction coefficient of the tracer with the bath,

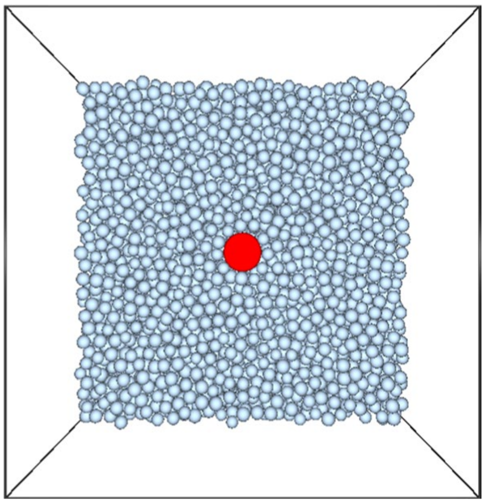


Fig. 3 Snapshot of the system with $N = 15{,}625$ particles. The tracer, three times larger than the bath particles, $a_t = 3a_b$, is marked in red, and the particles in front of it have been removed

γ_{eff}, is obtained from the average tracer velocity using the stationary state relation: $\mathbf{F}_{\text{ext}} = \gamma_{\text{eff}}\langle \mathbf{v}\rangle$. A large number of trajectories are therefore needed to obtain reliable values of γ_{eff}. However, the tracer distorts the bath as it displaces, and since it is much larger than the bath particles, FSE can be present. In fact, due to the periodic boundary conditions, an array of particles is simulated with the lattice sparing equal to the box size. Starting from the Navier–Stokes equation, Hasimoto [26] showed that the effective friction coefficient measured by an array of particles in an incompressible Newtonian fluid depends on the lattice spacing, L, as:

$$\frac{1}{\gamma_{\text{eff}}} = \frac{c}{L} + \frac{1}{\gamma_{\infty}} \tag{3}$$

where c is a constant that depends on the structure of the array, and γ_{∞} is the effective friction coefficient measured by an isolated particle [26]. Following this theoretical result, γ_{∞} can be obtained running simulations with different system sizes, L, in order to obtain $\gamma_{\text{eff}}(L)$, and extrapolate linearly to $1/L \to 0$. Note that changing the system size implies changing the number of particles because the volume fraction is constant. Figure 4 shows the results of γ_{eff} for seven system sizes, with the number of particles ranging from $N = 216$ to 15,625. The inverse friction coefficient is indeed linear for small systems, but deviates for $1/L \to 0$, due to the approximations in the theoretical model.

The full analysis of the finite size effects in the system necessitates a large number of simulations or tasks of (i) systems with different number of particles, N_i with $1 \leq i \leq I$, and (ii) a large number of trajectories (Q_i) for every system size (N_i), requiring (iii) solving N_i equations of motion repeatedly for each trajectory.

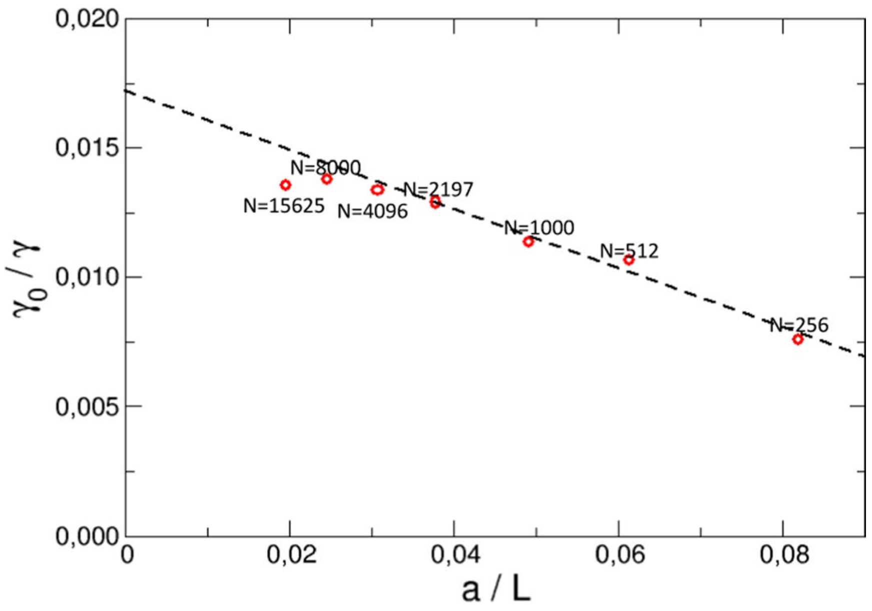


Fig. 4 Inverse friction coefficient vs. inverse length of the simulation box for a system with a volume fraction of $\phi = 0.50$, and a tracer of size $a_t = 3a_b$ pulled with a force $F = 2.5\,k_{\text{B}}T/a_b$. The labels indicate the number of particles used in every simulation. The number of trajectories analyzed for every point is 500

Therefore, the computational requirements of AMM models are very high which are provided by modern multi-GPU clusters. The model exhibits several parallelism levels, which allows the appropriate exploitation of such heterogeneous clusters. Previous works focused on accelerating the computation of a single tracer trajectory (bottom parallelism level) on the GPU [27, 28]. However, to advance in this kind of models, it is necessary to run efficiently many simulations in parallel on heterogeneous clusters (the highest parallelism level).

This way, the model defines a set of $M = \sum_{i=1}^{I} Q_i$ tasks which compute every tracer trajectory. So, tracer trajectories can be computed in parallel on the CPU-cores and GPUs of a cluster. Every CPU-core (GPU) can execute the sequential code (CUDA code) to compute one tracer trajectory, and the set of tasks can be computed with the collaboration of all processing units (PUs) of the cluster, CPU-cores and GPUs (PUs). Consequently, to get an optimal exploitation of heterogeneous clusters of models AMM is necessary to solve the scheduling problem on unrelated parallel machines defined in Sect. 2.

5 Results

In this section, the above-mentioned strategies for load balancing (GenS, Cyclic, Greedy, STQ and DETQ) the case study in AMM are evaluated on a wide variety of heterogeneous clusters. For all the estimations and tests, the same problem is used: System sizes of N_i= 216, 512, 1000, 2197, 4096, 8000 and 15,625, with 250 trajectories of 500 time units (corresponding to 10^6 time steps). Four kinds of PUs have been considered:

$Core_1$:	1 core of Bullx R424-E3 Intel Xeon E5 2650 with 8GB RAM
GPU_1 :	NVIDIA Tesla M2070 GPUs (Fermi)
$Core_2$:	1 core of Bullx R421-E4 Intel Xeon E5 2620v2 with 64GB RAM
GPU_2 :	NVIDIA Kepler GK210 (NVIDIA K80)

The characteristics of the GPU devices are given in Table 1. From these PUs, seven test clusters have been defined (five heterogeneous clusters and two homogeneous ones) to evaluate the scheduling methods (see Table 2).

Two implementations have been considered to simulate every tracer trajectory: a sequential CPU version coded in Fortran and a GPU version implemented in ANSI C and CUDA. Moreover, a Python's multiprocessing module has been used to code the schedulers considered in the experimental evaluation.

Firstly, focusing the attention on the static policies the profiling stage has been carried out. Hence, an estimation of the runtime for all system sizes involved in the case study and the four kinds of PUs (CPU-cores and GPUs) was obtained (matrix $\mathcal{T}$) and shown in Table 3. AF stands for the acceleration factor of each kind of device versus the slowest device for each system size. These values are used in the Greedy strategy, as it was mentioned in the previous section. Let us remark that the execution time increases with N_i. Moreover, the use of GPU computing is not beneficial

Table 1 Characteristics of the GPU devices

	M2070 (GPU_1)	GK210 (GPU_2)
Peak performance (double prec.) (TFlops)	0.51	2.91
Peak performance (simple prec.) (TFlops)	1.03	8.74
Device memory (GB)	5.2	24
Clock rate (GHz)	1.2	0.82
Memory bandwidth (GBytes/s)	150	480
Multiprocessors	14	13
CUDA cores	448	2496
Compute capability	2.0	3.7

Table 2 PUs provided for every test cluster (A–F)

	$Core_1$	GPU_1	$Core_2$	GPU_2	K
A	14	2			16
B	28	4			32
C	28	8			36
D	56	8			64
E	56	8	10	2	76
F		8			8
G	64				64

Table 3 Total execution time (in seconds) for seven tasks sizes (N_i). $t_{GPU1,i}$ ($t_{GPU2,i}$) and $t_{CPU1,i}$ ($t_{CPU2,i}$) columns identify the runtime to compute a single trajectory on a GPU of kind 1 (2) and a CPU-core of kind 1 (2)

N_i	$s = 1$	$s = 2$	$s = 3$	$s = 4$				
	$t_{GPU1,i}$	$t_{CPU1,i}$	$t_{GPU2,i}$	$t_{CPU2,i}$	AF_1	AF_2	AF_3	AF_4
216	1580	790	1406	101	1.0	2.0	1.1	15.6
512	1785	1860	1714	507	1.0	1.0	1.1	3.7
1000	2240	3715	2030	2319	1.7	1.0	1.8	1.6
2197	2930	8710	2112	5315	3.0	1.0	4.1	1.6
4096	4450	18,065	3235	10,465	4.1	1.0	5.6	1,7
8000	7650	43,080	4587	24,427	5.6	1.0	9.4	1.8
15,625	12,050	11,3940	10,043	63,788	9.5	1.0	11.3	1.8

AF_s, with $1 \leq s \leq 4$, are the acceleration factors of every kind of device versus the slowest one for each N_i

to accelerate microrheology problems when N_i is low. However, when $N_i \geq 1000$, GPUs increase the performance.

Secondly, focusing our attention on the second stage of our methodology, Table 4 shows the estimated parallel runtime ($C_{\max}$) in hours. Analyzing the homogeneous platforms (F and G), it is observed that GenS achieves the best makespan, equaling or improving the other strategies by 3%. So, for these platforms, the advantages of GenS are maintained although they are not very relevant. But for all the

Table 4 Makespan, in hours, for each strategy for cases exposed in Table 2

	Heterogeneous					Homogeneous	
	A	B	C	D	E	F	G
STQ	489.6	244.8	184.8	129.6	108.0	**283.2**	211.2
DETQ	412.8	223.2	141.6	122.4	103.2	**283.2**	211.2
Greedy	504.0	261.6	177.6	136.8	112.8	290.4	211.2
Cyclic	844.8	422.4	369.6	211.2	211.2	290.4	211.2
GenS	**410.4**	**206.4**	**139.2**	**102.5**	**79.2**	**283.2**	**206.4**

The scheduling scheme that obtains the best performance is marked in bold

heterogeneous platforms (A–E), the experimental results of GenS are significantly better than the other approaches, and this improvement is more evident as the heterogeneity and size of the platform increase. The Cyclic strategy always has the worst runtime by far, STQ obtains reasonable runtime close to the DETQ (the second best one), and the Greedy, although it improves the cyclic one, has large makespan.

It is relevant to underline that due to the non-deterministic behavior of GenS, it has been executed 10 times in order to check its robustness and the dispersion of the results has been less than 0.15%. Therefore, we can remark the high robustness of the GenS solution.

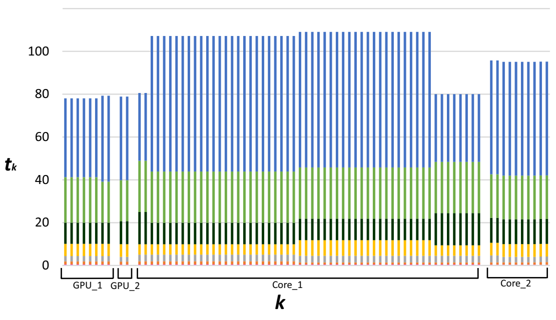
To demonstrate that a simple random search is not competitive with respect to GenS, it has been executed during a time interval significantly greater than the GenS runtime to solve the same problem. Results have shown that the GenS overcomes the random search in terms of makespan. For the sake of clarity, this study has not been included.

Let us now focus our attention on the heterogeneous cluster with more resources, E. Figure 5 (a–e) shows the runtime for every device from each strategy, and (f) shows the percentage of task sizes in every platform scheduled by the GenS (the colors show the task size, as given in the legend). In the STQ strategy (a), large tasks that consume a lot of time are computed on the CPU-cores; meanwhile, the GPUs are inactive, causing important imbalances with large makespans. In the DETQ strategy (b), the number of heavy tasks that come to the CPU-cores is not so important (the $Core_2$ does not take any, for example) and therefore the makespan is reduced, but there are still large imbalances. The Cyclic strategy (d), as the distribution is made without taking into account the heterogeneity of the platform, is the worst one. The Greedy strategy is also far from the optima. GenS tries to fit the heterogeneity of the tasks and the hardware of the platform, so its evolution makes the most powerful PUs compute the largest tasks (see that the $Core_2$ has more large tasks than $Core_1$) and being the less powerful devices those that are in charge of the light ones Fig. 5f. If we analyze the unbalance among the different platforms in Fig. 5e, we can conclude that the GenS solution is not far from the optimum since all devices finish almost at the same time (a makespan of 79.2 h).

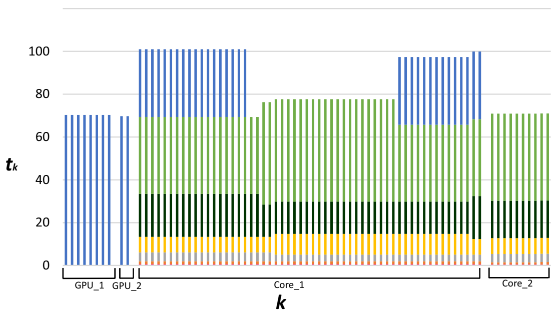
So, the GenS scheduling obtains the minimum estimation of parallel runtime. Then, the next step is to analyze the parallel executions on the test clusters (stage 3)

Table 5 Estimated (stage 2) and experimental makespan (stage 3), in hours, obtained by GenS scheduling, for heterogeneous clusters D and E from Table 2

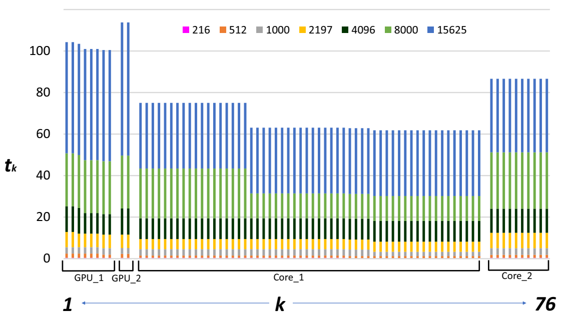
	Estimated		Experimental	
	D	E	D	E
C_{Max}	102.5	79.2	105.4	82.5
C_{Min}	101.8	76.4	104.4	79.4



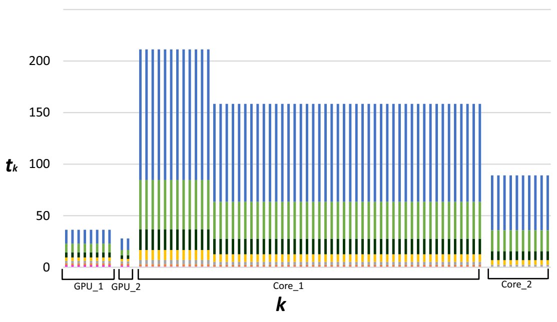
(a) STQ



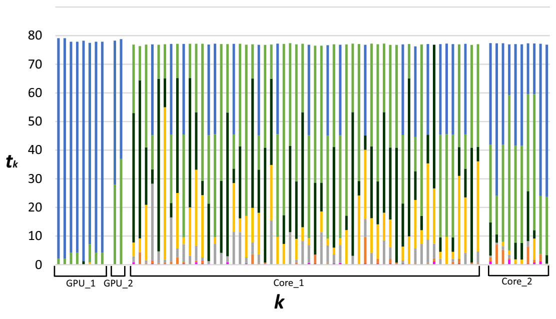
(b) DETQ



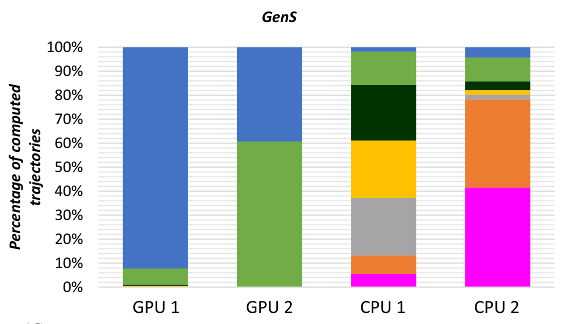
(c) Greedy



(d) Cyclic



(e) GenS



(f) Percentage of task of every type of PU solved by the GenS

Fig. 5 Figure **a–e** shows the runtime, in hours, for all the devices of every strategy on cluster E. Each column *k*, shown as several stacked bars, corresponds to a device. A stacked bar represents the time spent by device *k* to compute the tasks of size *i* assigned to the device, so the entire column represents its total runtime. Figure **f** shows the percentage of the sizes of the tasks scheduled by GenS on each platform

to verify that GenS estimations are realistic. Real executions on the clusters D and E (the clusters which have the largest number of devices) have been tested. Table 5 shows the estimated makespan by GenS, in hours, in comparison with executions using GenS scheduling on both clusters. Analyzing the execution time, it can be concluded that the estimation of the GenS is close to the makespan of the real experimentation. Experimental runtime is a little larger than the predicted one because GenS estimation does not take into account the runtime to prepare and to send a task to the corresponding machine and also the contention among the PUs in the clusters.

Therefore, using GenS to schedule a model composed by heterogeneous tasks on a heterogeneous cluster has resulted in an important reduction of the impracticable runtime of the previous versions of such model. For the study case, if the set of simulations is computed on a $Core_1$ the estimated sequential runtime would be 13205, 6 hours. Then, an acceleration factor of ×129 (×167) is achieved on cluster D (on cluster E) using the GenS scheduling.

6 Conclusion

In this work, the scheduling of heterogeneous tasks on unrelated parallel machines has been studied. An approach for distributing the workload in a near-optimal way based on a Genetic Algorithm (GenS) has been analyzed. GenS has been comparatively evaluated with respect to other schedulers using a real problem from the field of statistical mechanics (active microrheology model) as a case study. The goal of such model is the computation of the effective friction coefficient of complex fluids where Finite Size Effects are dominant. The computational cost for these models is huge because they are based on statistical analysis of the dynamics of a tracer particle for several system sizes. Therefore, the use of appropriate scheduling approaches on heterogeneous clusters has been a key to strengthen the applicability of these models.

Experimental results have shown that GenS achieves a near-optimal load balance, even when the cluster supplies a large and heterogeneous set of processing units, outperforming other studied strategies. GenS improves the performance with respect to the second fastest scheduling (DETQ) up to 23.56% on the cluster E (the highest heterogenous one). Thus, the advantages of GenS are more relevant as the cluster heterogeneity increases.

Only the evolution of GenS has allowed to define the assignment task/ processing-unit according to the load-of-task/ computational power optimally for highly heterogeneous tasks and processing units. This way, all processing units finish their computation almost simultaneously. A suitable definition of the operators and individuals involved in the Genetic Algorithm has been relevant to achieve these scheduling results.

The main contribution of this work has been to design and to provide a scheduling software for efficiently distributing a set of independent tasks varying in cost on heterogeneous processing units (https://github.com/2forts/GENS). Thus, this software can be useful for all problems which can be modeled by scheduling on unrelated parallel machines beyond the case study.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science throughout Project RTI2018-095993-B-I00 and by the European Regional Development Fund (ERDF). F. Orts is supported by an FPI Fellowship (attached to Project TIN2015-66680-C2-1-R) from the Spanish Ministry of Education.

References

1. Hennessy JL, Patterson DA (2011) Computer architecture: a quantitative approach. Morgan Kaufmann, Burlington
2. Lenstra JK, Shmoys DB, Tardos E (1990) Approximation algorithms for scheduling unrelated parallel machines. Math Progr 46(3):259–271
3. Shmoys DB, Tardos E (1993) An approximation algorithm for the generalized assignment problem. Math Progr 62(3):461–474. https://doi.org/10.1007/BF01585178
4. Augonnet C, Thibault S, Namyst R, Wacrenier P (2011) StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. Concurr Comp Pract E 23(2):187–198
5. Luk C, Hong S, Kim H (2009) Qilin: exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO 42. ACM, New York, pp 45–55
6. Pea McCormick (2007) Scout: a data-parallel programming language for graphics processors. Parallel Comput 33(10):648–662
7. Chend Q, Guo M (2017) Task scheduling for multi-core and parallel architectures: challenges solutions and perspectives. Springer, Berlin
8. Cicuta P, Donald AM (2007) Microrheology: a review of the method and applications. Soft Matter 3:1449–1455
9. Puertas AM, Voigtmann T (2014) Microrheology of colloidal systems. J Phys Condens Matter 26(24):243101
10. Gehrke JC, Jansen K, Kraft SEJ, Schikowski J (2016) A PTAS for scheduling unrelated machines of few different types. In: SOFSEM 2016: Theory and Practice of Computer Science. vol. 9587 of Lecture Notes in Computer Science. Springer, Berlin, pp 45–55
11. Sels V, Coelho J, Dias AM, Vanhoucke M (2015) Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem. Comput Oper Res 53:107–117. https://doi.org/10.1016/j.cor.2014.08.002
12. Wang T, Liu Z, Chen Y, Xu Y, Dai X (2014) Load balancing task scheduling based on genetic algorithm in cloud computing. In: Proceedings of the 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. DASC '14. IEEE Computer Society, pp 146–152
13. Sharma H, Sekhon GS (2017) Load balancing in cloud using enhanced genetic algorithm. Int J Innov Adv Comput Sci 6(1):13–19
14. Adan J, Adan I, Akcay A, Van den Dobbelsteen R, Stokkermans J (2018) A hybrid genetic algorithm for parallel machine scheduling at semiconductor back-end production. In: Twenty-Eighth International Conference on Automated Planning and Scheduling
15. Al-Said IAM, Al-Saiyd N, Attia FT (2008) Multiprocessor scheduling based on genetic algorithms. In: The International Arab Conference on Information Technology (ACIT'2008)
16. Cappadonna FA, Costa A, Fichera S (2012) Three genetic algorithm approaches to the unrelated parallel machine scheduling problem with limited human resources. In: In Proceedings of the 4th International Joint Conference on Computational Intelligence (ECTA-2012). pp 170–175
17. Hou ES, Ansari N, Ren H (1994) A genetic algorithm for multiprocessor scheduling. IEEE Trans Parallel Distrib Syst 5(2):113–120
18. Jooyayeshendi A, Akkasi A (2015) Genetic algorithm for task scheduling in heterogeneous distributed computing system. Int J Sci Eng Res 6(7):1338–1345
19. Page AJ, Naughton TJ (2005) Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. In: 19th IEEE International Parallel and Distributed Processing Symposium. 6(7)
20. Kaiser T, Jegede O, Ferens K, Buchanan D (2013) A genetic algorithm for multiprocessor task scheduling. In: Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM), The Steering Committee of The World Congress in Computer Science, p 1
21. Vallada E, Ruiz R (2011) A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. Eur J Oper Res 211(3):612–622

22. Woodside CM, Monforton GG (1993) Fast allocation of processes in distributed and parallel systems. IEEE Trans Parallel Distrib Syst 2:164–174
23. Waigh TA (2016) Advances in the microrheology of complex fluids. Rep Prog Phys 79(7):074601
24. Dhont JKG (1996) An introduction to dynamics of colloids. Studies in interface science. Elsevier Science, Amsterdam
25. Orts F, Ortega G, Garzón EM, Puertas AM (2019) Finite size effects in active microrheology in colloids. Comput Phys Commun 236(1):8–14
26. Hasimoto H (1959) On the periodic fundamental solutions of the Stokes equations and their application to viscous flow past a cubic array of spheres. J Fluid Mech 5:317–328
27. Ortega G, Puertas AM, de Las Nieves FJ, Garzón EM (2016) GPU computing to speed-up the resolution of microrheology models. In: Algorithms and Architectures for Parallel Processing: Proceedings of ICA3PP Conference. Springer International Publishing, Cham pp 457–466
28. Ortega G, Puertas AM, Garzón EM (2017) Accelerating the problem of microrheology in colloidal systems on a GPU. J Supercomput 73(1):370–383

Affiliations

F. Orts[1] · G. Ortega[1] · A. M. Puertas[2] · I. García[3] · E. M. Garzón[1]

F. Orts
francisco.orts@ual.es

A. M. Puertas
apuertas@ual.es

I. García
igarciaf@uma.es

E. M. Garzón
gmartin@ual.es

[1] Informatics Department, University of Almería, ceiA3, Carretera Sacramento s/n, La Cañada, Almería, Spain

[2] Department of Applied Physics, University of Almería, Carretera Sacramento s/n, La Cañada, Almería, Spain

[3] Computer Architecture Department, Campus Teatinos, Universidad de Málaga, Málaga, Spain

2.2.4 Quantum Algorithms to compute the neighbour list of N-body Simulations

Title	Quantum Algorithms to compute the neighbour list of N-body Simulations
Authors	*E.F. Combarro, I.F. Rúa, F. Orts, G. Ortega, A.M. Puertas and E.M. Garzón*
Journal	Submitted and under review.
Year	2021

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, has helped creating the oracle circuit associated to this paper.

Quantum Algorithms to compute the neighbour list of N-body Simulations

E. Fernández Combarro[1], I. Fernández Rua[2], F. Orts[3], G. Ortega[3], A.M. Puertas[4], E.M. Garzón[3]

Abstract

One of the strategies to reduce the complexity of $N-$body simulations is the computation of the neighbour list. However, this list needs to be updated from time to time, with a high computational cost. This paper focuses on the use of quantum computing to accelerate such a computation. Our proposal is based on a well-known oracular quantum algorithm (Grover). We introduce an efficient quantum circuit to build the oracle that marks pairs of closed bodies, and we provide three novel algorithms to calculate the neighbour list under several hypotheses which take into account a-priori information of the system. We also describe a decision methodology for the actual use of the proposed quantum algorithms. The performance of the algorithms is tested with a statistical simulation of the oracle, where a fixed number of pairs of bodies are set as neighbours. A statistical analysis of the number of oracle queries is carried out. The obtained results indicate that our algorithms can clearly outperform the best classical algorithm in terms of oracle queries, when the density of bodies is low.

Keywords: quantum computing, quantum algorithm, neighbour list, N-body simulations

[1]Department of Computer Science, University of Oviedo, Spain Openlab, CERN, Switzerland
[2]Department of Mathematics, University of Oviedo, C/ Federico García Lorca 18, 33007, Oviedo, Spain
[3]Supercomputation-Algorithms Group, Department of Informatics, University of Almería, ceiA3, Ctra. Sacramento s/n, 04120, Almería, Spain
[4]Group of Complex Fluids Physics, Department of Applied Physics, University of Almería, Ctra. Sacramento s/n, 04120, Almería, Spain

2010 MSC: 00-01, 99-00

1. Introduction

The N-body problem is widely used in simulations in a large variety of fields, from material science, statistical physics, to astrophysics [1, 2, 3]. However, the high computational load of N-body simulations is well-known. When the number of particles, N, is not too large, the interactions can be computed by a brute-force approach, with complexity order $O(N^2)$ [1, 2, 4]. Nevertheless, when N increases it is necessary to reduce the complexity.

Barnes & Hut defined a hierarchical tree cells scheme to locate the particles and an algorithm to compute the interactions with a complexity of $O(Nlog(N))$. It is widely applied to a large number of long-range interactions ranging from stellar dynamical applications [5] to material science or molecular dynamics [1]. Moreover, an adaption of Barnes & Hut' scheme has also been simplified for the approximate computation of long-range forces between mutually interacting bodies with a complexity of $O(N)$ [6].

In the context of short-range interactions, the main approach to get a complexity of $O(N)$ is to define a neighbour list, where the interactions are only computed among neighbour particles. However, the neighbour list has to be updated after several time steps and its complexity is $O(N^2)$. The frequency of such computation can be reduced if the neighbourhood radius is optimized [2, 7].

Our interest is the acceleration of simulations related to N-body systems with short-range interactions by the fast computation of neighbour lists. This technique is commonly used in computer simulations in many different fields, such as phase equilibria, equilibrium or out-of-equilibrium molecular dynamics, or soft-matter systems [8]. Particularly in suspensions of macromolecules or colloids, the interaction among the particles is of a much shorter range than the radius or typical length, making the use of neighbour lists very convenient. This has allowed the experimental realization of the paradigmatic hard-sphere model, or the attractive square-well with controllable range, in addition to the Lennard-Jones potential typical of atoms of molecules.

Quantum computing [9] can be considered as a strategy to predictably accelerate these computationally expensive simulations. Quantum computing relies on the basic quantum principles of superposition and entanglement, which make it suitable for accelerating parallel and distributed applications and also for improving networks and communications.

Previous works exploit the quantum parallelism in many-body system simulations based on adiabatic quantum computation [10, 11, 12]. In contrast, this paper addresses the N-body simulations considering quantum circuit algorithms to accelerate the computation of neighbour lists. It is designed using Grover's Algorithm, the main oracular quantum search algorithm [9].

The aim of this paper is two-fold. Firstly, to propose several comprehensive solutions to the computation of the neighbour list with quantum computing under different alternative hypothesis. The algorithms proposed here are tested with a simplified oracle, where a fixed number of pairs of particles are set as neighbours. The circuits obtained from this study are freely available at https://github.com/2forts/qsec. Secondly, to set a decision methodology for the actual use of the proposed quantum algorithms. And, additionally, to set a design methodology for the development of quantum algorithms, taking into account a comprehensive design that supplies both algorithms and related circuits.

The manuscript is organized as follows. In Section 2 an overview about quantum computing is established. Section 3 is devoted to describing the three proposed quantum algorithms for finding pairs of close particles and the selection criteria. Furthermore, details about the oracle design as a reversible quantum circuit are discussed. In section 4 statistical simulations to test the proposed algorithms with a simplification of the oracle are carried out. Finally, the conclusions are presented.

2. Quantum Computing Background

Quantum computers have been considered a promising technology from its introduction to our days. These computers benefit from the special and counter-intuitive properties of quantum mechanics, like superposition. Superposition allows a qubit (a quantum bit, the basic unit of the quantum computers) to be in the states $|0\rangle$ and $|1\rangle$ simultaneously. Thanks to this feature, quantum computers can evaluate a function $f(x)$ at many values of x at once, what is known as quantum parallelism [9].

Since their introduction, quantum algorithms have outperformed classical ones in several problems. Grover and Shor algorithms are the two best-known examples. In fact, most of the current quantum algorithms are based on the methodology of one of these two [9]. Focusing on Grover's algorithm, it performs a search through an unstructured space, achieving a quadratic speedup with respect to classic search algorithms. Among other quantum properties, Grover's algorithm is based on the concepts of superposition and quantum parallelism to compute several evaluations of a function as one [13]. The algorithm obtains a solution with a certain probability, being necessary a minimum of iterations of the algorithm to get the solution with the desired probability. The estimation of the necessary number of iterations is one of the most important parts in the algorithm.

Grover's algorithm needs a black box oracle O as an input. This oracle has to check if a value x is (or not) a solution to the search problem. Therefore, to apply Grover's algorithms to a real problem it is necessary to build an oracle with the capacity to recognize if a given value is a valid solution to that problem. It is just as important to use the algorithm in the correct context, as it is to build an efficient oracle for it. The circuits paradigm is the most usual methodology to design and implement quantum algorithms, where an oracle based on the design of reversible quantum circuits is required. In the literature, it is a common practice to mathematically define an oracle for the problem. However, without a real implementation, the algorithm is not functional on a quantum computer or simulator.

So, the methodology widely used to design quantum applications involves the combination of: (1) the design of quantum algorithms based on well-known quantum procedures (for example, Grover) bearing in mind the statistical computation provided by them and (2) and the use of a particular reversible quantum circuit that implements the specific oracle use in such design. In this work we provide a whole design of quantum algorithms to compute the neighbour list.

In the rest of this paper, we introduce a quantum algorithm based on Grover's algorithm, showing that it involves fewer queries than classical alternatives. Moreover, we present the complete design of the oracle for our algorithm, ready for its use in quantum simulators.

3. Quantum algorithms for finding pairs of close particles

In this section, we propose three quantum algorithms that can be used to find all the pairs of particles that are closer than a given threshold distance. For this, we will assume, as it is customary in this kind of problem [13, 9], that we are given a quantum circuit implementing an oracle O such that

$$O(|x\rangle |0\rangle) = \begin{cases} |x\rangle |1\rangle & \text{if } x \text{ satisfies certain conditions} \\ |x\rangle |0\rangle & \text{otherwise} \end{cases}$$

Notice that this is a completely general situation and can be applied not only for the case of finding all the pairs of particles that are close (in which case $|x\rangle = |x_1\rangle |x_2\rangle$, with x_1 and x_2 indices of two particles), but to any setting in which we have to find all the elements in a set that satisfy a certain condition. This is closely related to the Coupon Collector Problem [14], that has been recently studied in a quantum context [15] but with an important difference: in general, we do not know how many pairs of particles are closer than the threshold, so we are not able to use the methods presented in that work. Another important feature is the fact that, for a given particle, the number of close particles is upper bounded by a constant independent of the total number of particles.

The availability of the oracle O allows us to use Grover's search algorithm [13], that will be central to our methods. It is important to note that the success probability of Grover's algorithm and the number of times it consults the oracle are completely determined by the number of elements ν in the set and by the number μ of *marked* elements (i.e., elements that satisfy the condition). For that reason, in our algorithms we will consider oracles $O = O_\nu^\mu$ that mark exactly μ elements from a set of size ν. This general setting allows us to consider two different situations: we can search among all the pairs of particles at once (i.e., $\nu = N^2$, and μ is the number of pairs of close particles) or we can fix one of the particles and search for the close ones (i.e., $\nu = N$, and μ is the number of close neighbour). This will prove useful in certain situations, as we explain below, but from the point of view of the analysis of our quantum algorithms we can consider both cases in just one abstract setting, with the only difference being the values of the parameters ν and μ.

3.1. Oracle Construction

In this subsection we discuss the construction of a quantum circuit implementing the oracle O for the particular case of marking pairs of particles that are below a given distance. In this paper, we will consider that all our algorithms use that circuit as an instantiation of the oracle. Therefore, we want to demonstrate the feasibility of building such an oracle.

A circuit implementing the oracle must return 1 if the distance between two particles i and j is less than or equal to a threshold value δ, and 0 otherwise. That procedure can be divided into two operations: the computation of the distances between i and j, and the comparison between that distance and δ. Additionally, as required in two of the proposed algorithms, we will need to modify the oracle O so that, once found a marked element x_0, it is excluded from being marked by a new oracle O':

$$O'(|x\rangle|0\rangle) = \begin{cases} |x\rangle|1\rangle & \text{if } x \text{ is marked and } x \neq x_0 \\ |x\rangle|0\rangle & \text{otherwise} \end{cases}$$

Focusing on the arithmetic part, the process supports some simplifications. On the one hand, it is possible to work with the squared distances. Therefore, the square root of the distances between particles is not necessary. Then, the distances can be computed using subtractors, adders, and squaring circuits. On the other hand, the comparison can be computed using a half comparator instead of a full comparator since it is only necessary to identify if the distance is, or is not, less than or equal to the threshold. Half comparators involve less resources than full ones. Focusing now on the modification proposed in the previous equation, it can be achieved by standard procedures, such as for instance the use of X gates and a multi-controlled Toffoli gates. We will repeatedly use these modifications of the original oracles in our algorithms.

It is important to note that this oracle will not provide any quantum advantage. However, even quantum circuits that does not provide quantum advantages can be useful as part of larger circuits if they involve an small number of resources [16]. In our case, the oracle must use the least possible number of resources to be efficiently used by our algorithms. In terms of quantum circuits, resource optimization is commonly measured using the number of involved qubits. It is also important to avoid the so-called garbage outputs: qubits that are not part of the result and whose value is not restored to the initial one, so they cannot be used in other circuits. A reduction in the number of operations (represented by the so-called quantum cost) is also desirable [17, 18].

Table 3.1 shows some of the most prominent adders, subtractors, squaring circuits, and half-comparators available in the literature. The table shows their quantum cost, their number of ancilla inputs, and the number of garbage outputs, according to the definitions given by Mohammadi et al. [17]. To carry out a complete analysis of the available circuits in the state-of-the-art is out of the scope of this article. However, we have studied a few selection of them in order

	Circuit	**Quantum cost**	**Ancilla inputs**	**Garbage outputs**
Adders and subtractors	[23] (full subtractors)	$6n$	n	0
	[23] (full and half subtractors)	$6n-2$	n	0
	[24] + [23]	$6n$	$n+1$	0
	[20](input carry) ($\overline{\overline{a}+b}$)	$18n-6$	2	0
	[20](input carry) ($a+\overline{b}+1$)	$16n-4$	2	0
	[20](no input carry) ($\overline{\overline{a}+b}$)	$16n-8$	1	0
	[25] ($\overline{\overline{a}+b}$)	$31n-15W(n)-15log(n)-6$	$5n/4$	0
	[26] ($a+\overline{b}+1$)	$30n-15W(n)-15log(n)-4$	$5n/4$	0
Squaring circuits	[27]	$36n$	$7n$	$7n$
	[28]	$35n$	$10n$	$10n$
	[29]	$36n$	$7n$	$13n$
	[30]	$38n$	$13n$	$13n$
	[21]	$32n$	$6n-3$	0
Half comparators	[31]	$O(n^2)$	$2n$	0
	[32]	$39n+9$	$6n+1$	0
	[18]	$18n+9$	$4n-3$	0
	[33]	$14n$	$4n-2$	0
	[34]	$28n$	2	0
	[20] ($\overline{\overline{a}+b}$)	$32n-18$	3	0
	[20] ($a+\overline{b}+1$)	$30n-10$	3	0
	[23] (full and half subtractors)	$12n$	$2n-3$	0
	[22]	$16n-8$	2	0

Table 1: Evaluation of most optimized circuits which can be used as part of the oracle O for the general n-digit case, in terms of quantum cost, ancilla inputs and number of garbage outputs.

to implement a functional oracle. We have followed the methodology described in [19] to measure and to test these circuits. We have chosen the best circuits of each category to build the oracle, prioritizing the absence of garbage outputs and the number of ancilla inputs since their optimization involves less qubits. In particular, we have built and tested a prototype of the oracle in ProjectQ simulator using the circuits proposed in [20] (computing $\overline{\overline{a}+b}$), [21], and [22]. The source code is freely available in https://github.com/2forts/qsec.

3.2. The algorithmic methodology

All our algorithms are based on the use of Grover's search [13]. This quantum algorithm allows, given an oracle O^{μ}_{ν} that marks μ elements from a set of size ν, to find, with high probability, a marked element with $O\left(\sqrt{\frac{\nu}{\mu}}\right)$ consults to the oracle, compared to the $\Omega\left(\frac{\nu}{\mu}\right)$ that would be needed with a classical algorithm. This means that there is a quadratic gap between the upper-bound of the quantum algorithm, and the lower-bound of the classical ones. We will exploit this quadratic speed-up to obtain algorithms that are asymptotically faster than any possible classical algorithm that also uses a black-box oracle. Namely, this allows to beat the $\Omega(N^2)$ bound for the search of pairs of closed particles, in a non-quantum setting. Because of the intrinsic probabilistic nature of quantum computing, our algorithms will provide a right answer with probability at least $1 - w$, where w is a chosen input parameter.

We first consider the situation in which the number of marked elements μ is known. This case will be rarely encountered in practice (when our algorithms are used to find the pairs of particles that are below a given threshold), but we present it here anyway for two reasons. First, it is closely related to the Quantum Coupon Collector Problem, that has recently attracted some attention [15]. Second, it will provide a useful benchmark for the more realistic algorithms we present later, as an ideal minimal bound on the number of oracle consults.

Since we are assuming that we know μ, we can simply run Grover's algorithm, checking every time if we have obtained a new marked element, until all of them have been found. However, since Grover's algorithm only returns a marked element with certain probability, there is no upper bound to the number of required oracle consults. For that reason, we propose first to compute a number R of Grover iterations that guarantees finding all marked elements with probability of failure at most w (see the details in Appendix A). The complete procedure is, then, the one presented in Algorithm 1.

```
Algorithm 1.
INPUT:
  • An oracle $O_\nu^\mu$ marking a known number of $\mu$ elements in
    a database of $\nu$ elements ($0 < \mu \leq \frac{\nu}{2}$).
  • A desired error bound probability $0 < w < 1$.
OUTPUT:
  • A set of $r$ marked database elements $L = \{x_1, \ldots, x_r\}$.
    With probability at least $1 - w$, we will have $r = \mu$.
PROCEDURE:
  1. Set $L = \emptyset$; $R = \left\lceil \frac{\log\left(\frac{w}{\mu}\right)}{\log\left(1-\frac{1}{2\mu}\right)} \right\rceil$
  2. FOR $l$ from 1 to $R$ do
     (a) Run Grover's algorithm with $\left\lceil \frac{\pi}{4}\sqrt{\frac{\nu}{\mu}} \right\rceil$ iterations
     (b) If a marked element $x$ is found, set $L = L \cup \{x\}$
     (c) If $|L| = \mu$ GO TO 3.
  3. Return $L$
```

In practice, however, μ will be unknown to us. This affects our application of Grover's search in two different ways. On the one hand, we can never be sure that we have already found all the marked elements and this affects the stopping conditions (cf. line 2(c) of Algorithm 1). On the other, we do not know what is the optimal number of iterations in Grover's algorithm (cf. line 2(a) of Algorithm 1). Of course, not knowing μ, also prevents us from computing R.

To overcome these difficulties, we adopt a strategy similar to the one proposed in [35]. For the number of iterations in Grover's search, we select a random number in $\{0, \ldots, \lfloor\sqrt{\nu}\rfloor - 1\}$. For the stopping condition, we compute a value R that will guarantee that if after R executions of Grover's search no marked element has been found, then the probability that indeed there are marked elements is below w, an error bound selected by the user. The mathematical derivation of R is given in Appendix A. Note that this bound is very conservative and that, in practice, errors much smaller than w will be usually obtained, as shown in the numerical simulations that we have conducted (see Section 4).

The complete procedure is described in Algorithm 2. Notice that in line 3(b), after a new element has been found, we modify the oracle so that this element is not considered again. For that, we use the construction of oracle the O' mentioned above (Subsection 3.1).

225

Algorithm 2.

INPUT:

- *An oracle* O_ν^μ *marking an* unknown *number of* μ *elements (upper bounded by a* known or estimated B*) in a database of* ν *elements* ($0 \leq \mu \leq B \leq \frac{3\nu}{4}$*).*
- *A desired error bound probability* $0 < w < 1$.

OUTPUT:

- *A set of* r *marked database elements* $L = \{x_1, \ldots, x_r\}$*. With probability at least* $1 - w$*, we will have* $r = \mu$*.*

PROCEDURE:

```
1. Set $L = \emptyset$; $R = \left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{B}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil$; $FOUND = FALSE$
2. FOR $l$ from 1 to $R$ do
   (a) Choose $j$ uniformly at random from the set $\{0, \ldots, \lfloor\sqrt{\nu}\rfloor - 1\}$
   (b) Run Grover's algorithm with $j$ iterations
   (c) If a marked element $x$ is found, set $FOUND = TRUE$; GO TO 3.
3. IF $FOUND = FALSE$, OUTPUT $L$
   ELSE
   (a) Set $L = L \cup \{x\}$; $FOUND = FALSE$
   (b) Eliminate $x$ from the list of marked elements by the oracle
   (c) GO TO 2.
```

Although Algorithm 2 gives an acceptable worst case asymptotic behaviour (cf. Table 2), the average number of oracle consults can be improved by using techniques similar to the ones used in [35]. This yield us to introduce a third algorithm to achieve such an improvement (Algorithm 3). Instead of always choosing the number of iterations of Grover's algorithm in a uniform way (see line 2(a) in Algorithm 2), we now increase the number of iterations, starting from 1, by a factor of $\frac{6}{5}$ (see Algorithm 3, line 3.(a)). This allows us to improve the behaviour in the average case, as shown in Table 2. We still need, however, a stopping condition that guarantees that the probability of missing some elements is less than w, leading to a worst case behaviour equivalent to that of Algorithm 2. The details of the analysis can be found in Appendix A.

Table 2 summarises the oracle query complexities of the three algorithms that we have proposed, where we suppose that, in general, μ is a function of ν.

Algorithm	Worst case	Average case
1	$O\left(\sqrt{\nu\mu}\log(\mu)\right)$	$O\left(\sqrt{\nu\mu}\log(\mu)\right)$
2	$O\left(\sqrt{\nu}\mu\log(B)\right)$	$O\left(\sqrt{\nu}(\log(B)+\mu)\right)$
3	$O\left(\sqrt{\nu}\mu\log(\nu)\right)$	$O\left(\sqrt{\nu}(\log(\nu)+\sqrt{\mu})\right)$

Table 2: Summary of query complexities (ν is the size of the database, μ is the number of marked elements, $B \leq \frac{3\nu}{4}$ is an upper bound on μ)

Algorithm 3.

INPUT:

- *An oracle O^{μ}_{ν} marking an* unknown *number of μ elements (upper bounded by a* known or estimated *B) in a database of ν elements ($0 \leq \mu \leq B \leq \frac{3\nu}{4}$).*
- *A desired error bound probability* $0 < w < 1$.

OUTPUT:

- *A set of r marked database elements $L = \{x_1, \ldots, x_r\}$. With probability at least $1 - w$, we will have $r = \mu$.*

PROCEDURE:

```
1. Set $L = \emptyset$; $m = 1$; $\lambda = \frac{6}{5}$; $R = 1$; $FOUND = FALSE$
2. FOR $l$ from 1 to $R$ do
   (a) Choose $j$ uniformly at random from the set $\{0, \ldots, \lceil m \rceil - 1\}$
   (b) Run Grover's algorithm with $j$ iterations
   (c) If a marked element $x$ is found, set $FOUND = TRUE$; GO TO 3.
3. IF $FOUND = FALSE$
   (a) IF $m = \sqrt{\nu}$, OUTPUT $L$.
       ELSE,
               set $m = \min\{\lambda m, \sqrt{\nu}\}$;
               $FOUND = FALSE$.
               IF $m = \sqrt{\nu}$, set $R = \left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{B}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil$
   (b) GO TO 2.
   ELSE
   (a) Set $L = L \cup \{x\}$; $m = 1$; $R = 1$; $FOUND = FALSE$
   (b) Eliminate $x$ from the list of marked elements by the oracle
   (c) GO TO 2.
```

3.3. The case of particle pairs

The general search methods presented in the previous subsection can be applied to the problem of determining all the particle pairs that are closer than a given threshold distance. In this paper, the number of close particles to a fixed one is upper bounded by a constant independent of the total number of particles, because of the characteristics of the physical problem (see Section 4). We will explore two possible instantiations.

The first one is to consider all possible pairs of particles and apply any of the three algorithms directly. In this case, we will have $\nu = N^2$, where N is the total number of particles, and μ represents the number of pairs of close particles. Provided some mild conditions are met (see Appendix B), we obtain the asymptotic complexities shown in Table 3

Algorithm	Worst case	Average case
1	$O\left(N\sqrt{\mu}\log\mu\right)$	$O\left(N\sqrt{\mu}\log\mu\right)$
2	$O\left(N\mu\log B\right)$	$O\left(N(\log B+\mu)\right)$
3	$O\left(N\mu\log N\right)$	$O\left(N(\log N+\sqrt{\mu})\right)$

Table 3: Query complexities in our particular problem, first instantiation: pairs of close particles ($N \geq 54$ is the number of particles, μ is the number of pairs of close particles, $B \leq 27N$ is an upper bound on μ)

In the second instantiation, we fix one particle and search, with any of the three proposed algorithms, for all the particles that are close to it. This can be helpful, as explained in detailed in the next subsection, when only a few of the particles have changed their positions and, thus, we only need to update their neighbour lists. If we consider α to be the number of particles with new positions, then the complexities of the algorithms are those given in Table 4. For the detailed analysis, which is based on the key fact that the number of closed particles to a fixed one is upper bounded by a constant independent of the total number of particles, see Appendix B.

Notice that several of the algorithms offer asymptotic complexities which

Algorithm	Worst case	Average case
1	$O\left(\sqrt{N}\alpha\log\alpha\right)$	$O\left(\sqrt{N}\alpha\log\alpha\right)$
2	$O\left(\sqrt{N}\alpha\log\alpha\right)$	$O\left(\sqrt{N}\alpha\log\alpha\right)$
3	$O\left(\sqrt{N}\log(N)\alpha\log\alpha\right)$	$O\left(\sqrt{N}\log(N)\alpha\log\alpha\right)$

Table 4: Query complexities in our particular problem, second instantiation: particles close to a fixed one ($N \geq 54$ is the number of particles, α is the number of particles to search for close neighbours)

can be, in the average or even in the worst case, better than those of any classical algorithm (which, necessarily, would have to make $\frac{N(N-1)}{2}$ or αN distance computations and comparisons). In fact, we will show in Section 4 that for a range of parameter values found in real-life problems, our algorithms can greatly reduce the number of oracle queries that need to be performed.

In the next subsection, we explain how the different choices of algorithm can be integrated in a decision procedure depending on the problem parameters and the evolution of the system.

3.4. The decision procedure

As we can see, the second and third algorithms are memory procedures in which the input oracle must be updated in order to keep track of found elements. The three algorithms can be combined with different input parameters in order to obtain the set of close pairs of N particles in the space. Since the particles are continuously moving in space, we propose a two-step dynamic programming strategy: first, looking for close particles among the set of all pairs; later on, looking for close particles to fixed ones, when the positions of particles change (i.e., an update methodology). One aspect to be considered is that Algorithm 3 performs uniformly better than Algorithm 2 in the average case. So, if desired, Algorithm 3 could be a substitute for Algorithm 2 in the alternatives given below.

First step: initialise the pairs of close particles

At this initial stage, the parameter ν is to be instantiated as N^2, and μ is the number of close pairs to be found. The choice of the algorithms is as follows:

- **If** μ is not known, then:
  - **If** μ is believed to be negligible in relation to the total number of pairs, use Algorithm 2 ($O(N)$ oracle calls in the worst case) with an estimated upper bound $B \leq 27N$ of μ.
  - **Else**, use Algorithm 3 with an estimated upper bound $B \leq 27N$ of μ ($O(N\sqrt{N})$ oracle calls in the average case).
- **Else** (μ is known), then:
  - **If** μ is negligible in relation to the total number of pairs, use Algorithm 1 (in the worst scenario, $O(N)$ oracle calls) or Algorithm 2 ($O(N \log N)$ oracle calls in the worst case) with $B = \mu$.
  - **Else**, use Algorithm 1 ($O(N\sqrt{N}\log N)$ oracle calls in the worst case) or Algorithm 3 with $B = \mu$ ($O(N\sqrt{N})$ oracle calls in the average case).

Second step: update the set of particles close to fixed ones

At this stage, the parameter ν is to be instantiated as N, the number of updated particles is α, and for a fixed particle, μ represents the number of close particles to be found.

The alternatives are the following:

1. If $\alpha \log \alpha$ is close to N, then backtrack to the first step.
2. Else, set $S = \left\lceil \frac{\log(\frac{w}{\alpha})}{\log(w)} \right\rceil$. Then:
   (a) If μ is known, then use Algorithm 1 S times for each of the α particles ($O(\sqrt{N}\sqrt{\alpha}\log \alpha)$ oracle calls in the worst case).
   (b) Else, use Algorithm 2 S times for each of the α particles ($O(\sqrt{N}\sqrt{\alpha}\log \alpha)$ oracle calls in the worst case).

4. Statistical simulation of the algorithms

In this section, the performance of the first-step algorithms introduced in Section 3 are tested in practical situations. A key aspect of the simulation is the oracle O, where the particle configuration should be fed into, and the use of Grover's search. For the purpose of testing the actual behaviour of algorithms $1-3$, the oracle is simplified notably, just taking into account the number μ of pairs of close particles, among the total number of N particles. The simulation will simply identify such a number of pairs. Since Grover executions in the algorithms are independent, we can directly simulate (because of the results in [35]) the running of the Grover steps by sampling from a Bernoulli distribution with success probability given by

$$\sin^2((2j+1)\theta)$$

where j is the number of Grover iterations, $\sin^2\theta = \frac{t}{\nu}$ and t is the number of marked elements (notice that $t = \mu$ for Algorithm 1, but in Algorithms 2 and 3 t starts at μ and is decreased in one unit with each found element). This means that we do not actually run the Grover steps: we simply simulate the success probability of such runs, instead. In the case of Algorithms 2 and 3 that is enough, because each successful run of Grover will find a different element (we eliminate the obtained ones from the oracle). For Algorithm 1, when the simulation shows that Grover has found a marked element, we sample uniformly from the set $\{1, 2, \ldots, \mu\}$ to determine the actual element that has been found.

In all cases, three values of μ are considered, $\mu = 40$, 80, and 150. This implies a mean number of neighbours per particle ranging from 2.3 to 0.08, which corresponds to some situations found in practice. For instance, in the canonical hard-sphere system, taking a threshold value for the center to center distance of $3a$, with a the particle radius, these mean number of neighbours are obtained volume fractions below.

For Algorithm 1, following the analysis of Appendix A, the bounds on the total number of iterations for different success probabilities are given in Tables 5, 6 and 7. These bounds, however, are shown to be very conservative once we take into account the actual results found in the simulations. In Tables 8, 9 and 10 we show the minimum, maximum, average and standard deviation of the number of oracle calls needed until all the pairs are found, across 10^6 repetitions of the algorithm. Notice that these values are much lower than those expected from the asymptotic analysis, even when we take into account the standard deviation.

Error bound w	# Calls 125 part.	# Calls 216 part.	# Calls 512 part.	# Calls 1000 part.
0.1	7632	15264	30528	61056
0.05	8512	17024	34048	68096
0.01	10560	21120	42240	84480
0.005	11440	22880	45760	91520
0.001	13488	26976	53952	107094

Table 5: Bounds on # of oracle calls for Algorithm 1 when $\mu = 40$

Error bound w	# Calls 125 part.	# Calls 216 part.	# Calls 512 part.	# Calls 1000 part.
0.1	12804	24541	48015	96030
0.05	14124	27071	52965	105930
0.01	17208	32982	64530	129060
0.005	18540	35535	69525	139050
0.001	21612	41423	81045	162090

Table 6: Bounds on # of oracle calls for Algorithm 1 when $\mu = 80$

In Table 11, we show the value of R for Algorithms 2 and 3 for $B = 27N$. . Again, these bounds prove to be extremely conservative. We have executed Algorithms 2 and 3 for 10^6 times with values of R taken from $\{5, 10, \ldots, 70\}$. The full results can be found in the supplementary material. In this section, we present only the data for the first value of R that successfully finds all the

Error bound w	# Calls	# Calls	# Calls	# Calls
	125 part.	216 part.	512 part.	1000 part.
0.1	19719	37247	72303	144606
0.05	21582	40766	79134	158268
0.01	25920	48960	95040	190080
0.005	27792	52496	101904	203808
0.001	32130	60690	117810	235620

Table 7: Bounds on # of oracle calls for Algorithm 1 when $\mu = 150$

Particles	Minimum	Maximum	Average	Standard deviation
125	928	12600	2749.08	790.33
216	1888	24224	5481.58	1575.03
512	3904	44928	10957.61	3150.78
1000	7552	86144	21909.18	6313.69

Table 8: Minimum, maximum, average and standard deviation of the number of iterations for 10^6 repetitions of Algorithm 1 when $\mu = 40$

Particles	Minimum	Maximum	Average	Standard deviation
125	1908	20064	4920.50	1243.43
216	3795	33833	9181.87	2318.84
512	7254	61650	17887.36	4516.25
1000	14940	131490	35743.77	9016.89

Table 9: Minimum, maximum, average and standard deviation of the number of iterations for 10^6 repetitions of Algorithm 1 when $\mu = 80$

Particles	Mininum	Maximum	Average	Standard deviation
125	3636	28665	8038.76	1819.60
216	6613	49691	14415.13	3266.95
512	12606	92532	27695.03	6265.49
1000	24354	180774	55391.35	12542.27

Table 10: Minimum, maximum, average and standard deviation of the number of iterations for 10^6 repetitions of Algorithm 1 when $\mu = 150$

particle pairs in all 10^6 experiments for a fixed value of μ. Since all these results can be quickly obtained from simulations alone, for other values of N, ν and μ, one can repeat experiments similar to the ones presented here in order to determine, before using an actual quantum computer, which algorithm is most suitable for the situation and what is the desirable value of R. In Tables 12 through 17 we show those results, including the value of R and the minimum, maximum, average and standard deviation of the number of oracle calls used by the algorithms.

We can see that, as it was the case with Algorithm 1, Algorithms 2 and 3, we achieve an error rate below one in a million for values of R much less than what Table 11 would lead to expect.

Error bound w	R 125 part.	R 216 part.	R 512 part.	R 1000 part.
0.1	37	39	41	44
0.05	39	42	44	46
0.01	45	47	50	52
0.005	47	50	52	54
0.001	53	55	58	60

Table 11: Number of repetitions for different error bounds in Algorithms 2 and 3 when $\mu = 40$

In Figures 1, 2 and 3, we compare the number of queries needed by the classical algorithm with the average number of queries made by Algorithms 1,

Particles	R	Mininum	Maximum	Average	Standard deviation
125	30	4275	11155	6966.10	679.77
216	30	8783	22207	13987.19	1364.44
512	30	17789	43981	28031.48	2729.62
1000	30	34156	90053	56105.27	5462.89

Table 12: Minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 when $\mu = 40$

Particles	R	Mininum	Maximum	Average	Standard deviation
125	20	2027	4928	3183.36	260.28
216	20	4255	10959	6742.70	528.98
512	20	8806	22982	13986.88	1067.27
1000	20	19203	43485	28652.95	2151.95

Table 13: Minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 when $\mu = 40$

Particles	R	Mininum	Maximum	Average	Standard deviation
125	30	8209	17179	12066.42	948.43
216	30	16616	35536	24232.50	1905.19
512	30	33531	69521	48549.50	3805.91
1000	30	66544	139891	97211.92	948.43

Table 14: Minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 when $\mu = 80$

Particles	R	Mininum	Maximum	Average	Standard deviation
125	20	2572	5504	3815.21	271.06
216	20	5832	11881	8242.92	552.67
512	20	12368	24762	17312.67	1117.62
1000	20	25475	50528	35718.52	2251.94

Table 15: Minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 when $\mu = 80$

Particles	R	Mininum	Maximum	Average	Standard deviation
125	35	15946	28345	21269.77	1288.21
216	35	31338	56721	42704.70	2586.14
512	35	63555	112327	85583.67	5176.96
1000	35	127876	226940	171312.89	10360.06

Table 16: Minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 when $\mu = 150$

Particles	R	Mininum	Maximum	Average	Standard deviation
125	20	3178	6341	4522.74	280.11
216	20	7495	13782	10012.76	572.83
512	20	15898	28518	21342.74	1160.47
1000	20	32971	57502	44433.08	2337.95

Table 17: Minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 when $\mu = 150$

2 and 3. Notice that, while the growth in the case of the classical algorithm is quadratic, for our algorithms it is linear for fixed values of μ. In fact, for the lowest values of μ, the average number of queries of all our algorithms is lower than the number of queries performed by the classical algorithm. For bigger values of μ (80 and 150), the classical algorithm beats some of the quantum algorithms for low number of particles (125 and 216) but for the simulations with 512 and 1000 particles, our algorithms are always better (and the speed-up increases with the number of particles). In fact, Algorithm 3 was always better than the classical algorithm for all the cases under study.

These data show that our algorithms can clearly outperform the best classical algorithm in terms of oracle queries when the density of particles is low (μ is low or ν is high). Thus, once robust quantum hardware is available, these methods, especially Algorithm 3, may be of use in practical situations, where the density is usually low, a situation in which our algorithms show their better performance.

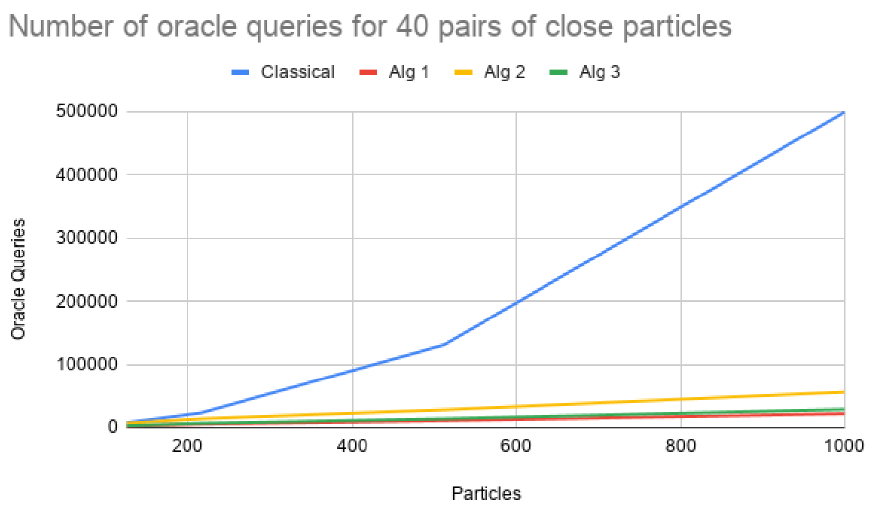


Figure 1: Comparison of the number of oracle queries of the different algorithms when $\mu = 40$

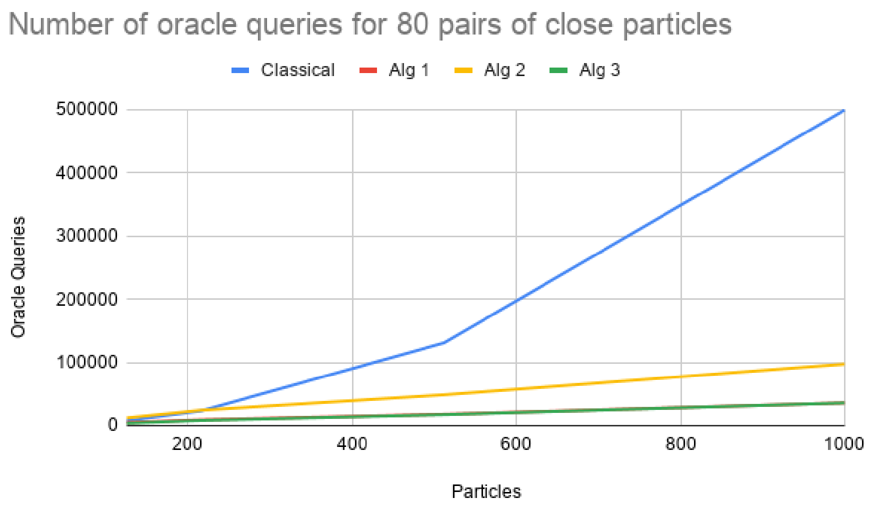


Figure 2: Comparison of the number of oracle queries of the different algorithms when $\mu = 80$

5. Conclusions

The focus of this work has been on the use of quantum computing to efficiently calculate the neighbour list in the context of N-body simulations. A quantum algorithm, based on oracle procedures (Grover) has been considered to carry out the whole proposal. The oracle has been designed with efficient reversible circuits that identify if pairs of bodies are neighbours or not. A prototype of the oracle has been developed in ProjectQ simulator based on the circuits proposed in [20, 21, 22] and it is available at https://github.com/2forts/qsec. Three quantum algorithms have been designed to get the pairs of neighbour particles from the information provided by the oracle. They can be combined in a two-step procedure for achieving such an objective: first, looking for pairs of close particles; second, updating the neighbour list of a small number of particles that move beyond a certain threshold. The actual combination of the algorithms has been described in a decision procedure, that aims to provide the best algorithm for each possible situation.

The asymptotic analysis of every algorithm has been justified from a the-

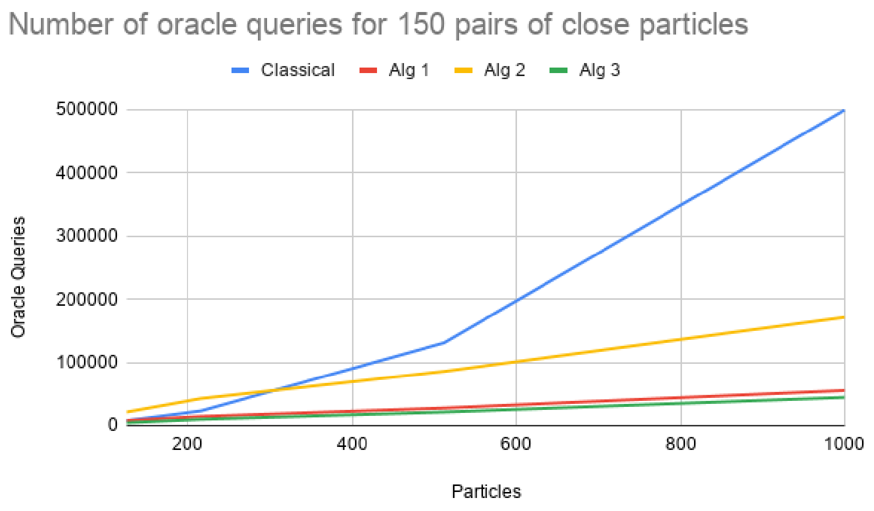


Figure 3: Comparison of the number of oracle queries of the different algorithms when $\mu = 150$

oretical point of view. A statistical simulation of the oracle O in combination with the algorithms has been considered to test their statistical behavior for μ pairs of close particles, among N particles.

After 10^6 repetitions of the algorithms, the developed test has evaluated the minimum, maximum, average, and standard deviation of the number of oracle calls needed until all the pairs were found. The obtained values have been much lower than those expected from the asymptotic analysis.

Thus, once robust quantum hardware is available, these methods, especially Algorithm 3, may be of use in practical situations, where the density is usually low, a situation in which our algorithms have shown their best performance.

Acknowledgments

This work has been partially supported by the Spanish Ministerio de Ciencia and FEDER (Projects No. PGC2018-101555-B-I00, RTI2018-095993-B-100, RTI2018-098085-B-C44), by UAL/CECEU/FEDER (Projects No. UAL18-TIC-A020-B and UAL18-FQM-B038-A), by the Ministry of Economy, Industry and Competitiveness from Spain/FEDER under grant MTM-2017-83506-C2-2-P, and by he Regional Ministry of the Principality of Asturias under grants FC-GRUPIN-IDI/2018/000193 and FC-GRUPIN-IDI/2018/000226.

Appendix A. Mathematical proof of the asymptotic behaviour of the proposed quantum algorithms

Algorithm 1

Given a database of ν unsorted elements and an oracle that detects $\mu = \mu(\nu)$ marked elements, Algorithm 1 provides a method that finds all marked elements with a bounded probability error, based on a repeatedly use of Grover's algorithm. We shall require that, for all ν, $0 < \mu(\nu)$. We will also assume that the sequence $\mu(\nu)$ has a limit, when $\nu \to \infty$.

Grover's algorithm provides, with $O\left(\sqrt{\frac{\nu}{\mu(\nu)}}\right)$ oracle calls, a success probability greater or equal than $\delta(\nu) := 1 - \frac{\mu(\nu)}{\nu}$, i.e., $\delta(\nu) := P(\text{finding a marked element out of the } \mu(\nu))$ [35, Section 3]. Assuming that $\mu(\nu) \leq \frac{\nu}{2}$, for all ν, we have a uniformly bounded success probability $\delta(\nu) \geq \frac{1}{2}$. Because such an algorithm does not distinguish between marked elements, we have that

$$P_i(\nu) := P(\text{finding the } i-\text{th marked element out of the } \mu(\nu)) = \frac{\delta(\nu)}{\mu(\nu)} \geq \frac{1}{2\mu(\nu)}$$

for all $i = 1, \ldots, \mu(\nu)$, and for all ν. We want to independently repeat the search $R = R(\nu)$ times and estimate the probability $P'(\nu)$ of not finding all marked elements. Namely,

$$P'(\nu) := P(\text{not finding all marked elements in } R(\nu) \text{ experiments})$$

$$= P(\text{not find. the first elem. in } R(\nu) \text{ exp.} \vee \ldots \vee \text{not find. the } \mu(\nu)-\text{th elem. in } R(\nu) \text{ exp.})$$

$$\leq \mu(\nu)\left(1 - \frac{1}{2\mu(\nu)}\right)^{R(\nu)}$$

In order to obtain a bounded algorithm, we require that such a probability is less than some $w < 1$, for all ν. This yields $\mu(\nu)\left(1 - \frac{1}{2\mu(\nu)}\right)^{R(\nu)} \leq w$ or, equivalently,

$$R(\nu) \geq \frac{\log\left(\frac{w}{\mu(\nu)}\right)}{\log\left(1 - \frac{1}{2\mu(\nu)}\right)}$$

Taking $R(\nu)$ as $\left\lceil \frac{\log\left(\frac{w}{\mu(\nu)}\right)}{\log\left(1-\frac{1}{2\mu(\nu)}\right)} \right\rceil$, we have that $R(\nu) = O\left(\mu(\nu)\log(\mu(\nu))\right)$, and the procedure requires an overall number of $O\left(\sqrt{\nu\mu(\nu)}\log(\mu(\nu))\right)$ oracle calls.

# Marked elements	#Iterations	#Orac. calls per it.	Total # oracle calls
$\mu(\nu)$	$O\left(\mu(\nu)\log(\mu(\nu))\right)$	$O\left(\sqrt{\frac{\nu}{\mu(\nu)}}\right)$	$O\left(\sqrt{\nu\mu(\nu)}\log(\mu(\nu))\right)$

Table A.18: Summary of Algorithm 1

The main obstacles to a practical application of this methodology are the requirements on $\mu(\nu)$, namely it has to be *known* and satisfy $0 < \mu(\nu) \leq \frac{\nu}{2}$, for all ν. Moreover, the correctness of the asymptotic analysis is conditioned to the sequence $\mu(\nu)$ having a limit. Since $\mu(\nu)$ is not always known, Algorithms 2 and 3 give two practical approaches based on Grover's algorithm with a random number of iterations. In both cases, an algorithm with memory and an appropriate time-out is taken.

Algorithm 2

This algorithm consists in a direct randomisation of the number of Grover's iterations of Algorithm 1. The list L keeps track of marked elements already found (a memory list), and the number $R = R(\nu)$ of times that Grover's search is repeated has to be taken so that the algorithm has a bounded success probability. This time we shall require that, for all ν, $0 < \mu(\nu) \leq \frac{3\nu}{4}$, and that the sequence $\mu(\nu)$ has a limit, when $\nu \to \infty$.

Let us consider the correctness of the second step in a single iteration of the algorithm. In such a step, the number of marked elements by the oracle is $0 \leq t \leq \frac{3\nu}{4}$. When $t = 0$, the algorithm forces (in the third step) OUTPUT L with no new elements added to the list L, and the output is right. On the other hand, when $t > 0$, because of Lemma 2 and the proof of Theorem 3 in [35], the probability of finding a marked element is $\delta(\nu) \geq \frac{1}{4}$, with $O\left(\sqrt{\nu}\right)$ oracle calls, so the overall probability of finding a marked element is $1 - (1-\delta(\nu))^{R(\nu)} \geq 1 - \left(\frac{3}{4}\right)^{R(\nu)}$.

Since the second step must be independently repeated $\mu(\nu)+1$ times for the algorithm to succeed (the last iteration is the one forcing the output), the probability $P'(\nu)$ of not finding all marked elements is $P'(\nu) := 1-\left(1-(1-\delta(\nu))^{R(\nu)}\right)^{\mu(\nu)} \leq 1-\left(1-\left(\frac{3}{4}\right)^{R(\nu)}\right)^{\mu(\nu)}$ which, in order to obtain a bounded algorithm, is required to be less than some $w < 1$, for all ν. This yields

$$R(\nu) \geq \frac{\log\left(1-(1-w)^{\frac{1}{\mu(\nu)}}\right)}{\log\left(\frac{3}{4}\right)}$$

Taking $R(\nu)$ as $\left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{\mu(\nu)}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil$, we have that $R(\nu) = O\left(\log(\mu(\nu))\right)$, and the procedure requires an overall number of $O\left(\sqrt{\nu}\mu(\nu)\log(\mu(\nu))\right)$ oracle calls. Of course, since $\mu(\nu)$ is assumed to be unknown, in practice we might know an upper bound $B(\nu)$ of $\mu(\nu)$ (in the worst case we can always choose $B(\nu) = \frac{3\nu}{4}$). This allows to take $R(\nu) = \left\lceil \frac{\log\left(1-(1-w)^{\frac{1}{B(\nu)}}\right)}{\log\left(\frac{3}{4}\right)} \right\rceil = O(\log(B(\nu))$ and the overall asymptotic complexity is $O\left(\sqrt{\nu}\mu(\nu)\log(B(\nu))\right)$.

#Step 2 iterations	#Iterations in Step 2	#Orac. calls per it.	Total # oracle class
$\mu(\nu)+1$ (output iter.)	$O\left(\log(B(\nu))\right)$	$O\left(\sqrt{\nu}\right)$	$O\left(\sqrt{\nu}\mu(\nu)\log(B(\nu))\right)$

Table A.19: Summary of Algorithm 2: worst case

In this algorithm, it is also interesting to analyse the average number of oracle queries. Since the probability of finding an element in any of the Grover executions of the loop of step 2 is at least $\frac{1}{4}$, the average number of queries on each execution of step 2 is less than $4\frac{\sqrt{\nu}}{2} = 2\sqrt{\nu}$ when there are still marked elements to be found. We need to add to that the number of queries of the output iteration (when all elements have already been found) to obtain an average number of queries which is $2\sqrt{\nu}\mu(\nu) + O\left(\sqrt{\nu}\log(B(\nu))\right) = O\left(\sqrt{\nu}(\log(B(\nu)) + \mu(\nu))\right)$.

#Step 2 iterations	#Iterations in Step 2	#Orac. calls per it.	Total # oracle class
$\mu(\nu) + 1$ (output iter.)	4 or $O\left(\log(B(\nu))\right)$	$\frac{\sqrt{\nu}}{2}$ or $\sqrt{\nu}$	$O\left(\sqrt{\nu}(\log(B(\nu)) + \mu(\nu)\right)$

Table A.20: Summary of Algorithm 2: average case

The main obstacles to a practical application of this methodology are: the requirements on $\mu(\nu)$, as it has to satisfy $0 < \mu(\nu) \leq \frac{3\nu}{4}$, for all ν; the asymptotic behaviour of the algorithm, which is worst than in the straightforward approach; the need of a continuous oracle update. The main advantages are that $\mu(\nu)$ is now not required to be known, and that the sequence $\mu(\nu)$ is not required to have a limit, when $\nu \to \infty$.

Algorithm 3

This alternate algorithm is a variation of the previous one, based on [35], and it consists in two stages. In the first one, the parameter m increases from 1 to $\sqrt{\nu}$ by a factor of λ. In each iteration, Grover's algorithm is only run once. When the *critical* stage is reached (i.e., when $m = \sqrt{\nu}$), the algorithm behaves exactly as the previous one. Since the algorithm never outputs before reaching the critical stage, the error probability is bounded as above. The difference here consists on the number of oracle calls. In the worst case, the algorithm performs the number of calls of the previous algorithm plus the oracle calls of the noncritical stage, but this latter number is $O\left(\sqrt{\nu}\log(\nu)\right)$, since $O(\log(\nu))$ iterations are needed to reach the critical stage. So the overall complexity of the worst case is $O\left(\sqrt{\nu}\mu(\nu)\log(\nu)\right)$.

Again, the average number of queries can be substantially lower than that. Indeed, from Theorem 3 in [35], when there are $t > 0$ marked elements to be found, the average number of oracle queries that our algorithm needs to perform in order to find one of them is $O\left(\sqrt{\frac{\nu}{t}}\right)$. Hence, the average number of

#Step 2 iterations	#Iter. to reach the critical stage	#Orac. calls per it.	Total # oracle class
$\mu(\nu)+1$ (output iter.)	$O\left(\log(\nu)\right)$	$O\left(\sqrt{\nu}\right)$	$O\left(\sqrt{\nu}\mu(\nu)\log(\nu)\right)$
#Step 2 iterations	#Iter. in Step 2 (critical stage)	#Orac. calls per it.	Total # oracle class class
$\mu(\nu)+1$ (output iter.)	$O\left(\log(B(\nu))\right)$	$O\left(\sqrt{\nu}\right)$	$O\left(\sqrt{\nu}\mu(\nu)\log(B(\nu))\right)$

Table A.21: Summary of Algorithm 3: worst case (noncritical and critical stages)

queries is $O\left(\sum_{t=1}^{\mu(\nu)}\sqrt{\frac{\nu}{t}}\right)+O\left(\sqrt{\nu}\log(\nu)\right)+O\left(\sqrt{\nu}\log(B(\nu))\right)=O\left(\sqrt{\nu\mu(\nu)}\right)+O\left(\sqrt{\nu}\log(\nu)\right)=O\left(\sqrt{\nu}(\log(\nu)+\sqrt{\mu(\nu)})\right)$, because $B(\nu)=O(\nu)$ (see Table A.22).

#Step 2 iterations	#Orac. calls per it.	Total # oracle class
$t=1,\dots,\mu(\nu)$	$\sqrt{\frac{\nu}{t}}$	$O\left(\sqrt{\nu\mu(\nu)}\right)$
1 (output iter.)	$\sqrt{\nu}\log(\nu)$ (noncritical) $+\sqrt{\nu}\log(B(\nu))$	$O(\sqrt{\nu}\log(\nu))$

Table A.22: Summary of Algorithm 3: average case

The obstacles to a practical application of this algorithm are mostly the ones of the previous one. However, although its asymptotic number of calls is never smaller than the algorithm above, its average number of queries can be better in practice (this has been observed in simulations) . In fact, even though the worst case query complexity is worse than that of the first algorithm proposed, the average number of queries is better when $\log(\nu)+\sqrt{\mu(\nu)}$ is $o(\sqrt{\mu(\nu)}\log(\mu(\nu)))$.

Summary of complexities

In Table A.23, we provide a table that summarises the complexities of the three algorithms that we have proposed.

Algorithm	Worst case	Average case
1	$O\left(\sqrt{\nu\mu(\nu)}\log(\mu(\nu))\right)$	$O\left(\sqrt{\nu\mu(\nu)}\log(\mu(\nu))\right)$
2	$O\left(\sqrt{\nu}\mu(\nu)\log(B(\nu))\right)$	$O\left(\sqrt{\nu}(\log(B(\nu))+\mu(\nu))\right)$
3	$O\left(\sqrt{\nu}\mu(\nu)\log(\nu)\right)$	$O\left(\sqrt{\nu}(\log(\nu)+\sqrt{\mu(\nu)})\right)$

Table A.23: Summary of query complexities ($B(\nu) \leq \frac{3\nu}{4}$ is an upper bound of $\mu(\nu)$)

Appendix B. Rationale behind the decision procedure

As mentioned in the text, the decision procedure for the determination of pairs of close particles consists in two steps. First, look for close particles among the set of all pairs. Second, look for close particles to a fixed one, when the positions of particles change (i.e., an update methodology). In each case, any of the three methods above can be potentially used. Next we explain the rationale behind our proposal.

First step: look directly for pairs of close particles

In this case $\nu = N^2$, and the required bounds on $\mu(N^2)$ are always satisfied when the number of particles is $N \geq 54$ (for the first algorithm) or $N \geq 36$ (for the second and third ones), because the characteristics of the physical problem (see Section 4). However, for smaller sizes of the problem and particularly small values of $\mu(N^2)$ the algorithms could still work. The assumption that $\mu(N^2)$ has a limit, as $N^2 \to \infty$, is realistic since the density is fixed, namely, the ratio of number of particles to available space is constant. Therefore, the more particles we have, the more chances of having pairs of close particles, i.e., it seems realistic assuming that $\mu(N^2)$ is non-decreasing, and so it has a limit. The main obstacle for using the first algorithm is the need of a knowledge of the actual value of $\mu(N^2)$. The asymptotic number of oracle calls of each algorithm is given in Table B.24

Depending on the actual $\mu(N^2)$, we will have different complexities. For instance, it has been noticed in practice that sometimes the number of close pairs of *distinct* particles is small in relation to the total number of pairs. This

Algorithm	Worst case	Average case
1	$O\left(N\sqrt{\mu(N^2)}\log\big(\mu(N^2)\big)\right)$	$O\left(N\sqrt{\mu(N^2)}\log\big(\mu(N^2)\big)\right)$
2	$O\left(N\mu(N^2)\log\big(B(N^2)\big)\right)$	$O\left(N(\log\big(B(N^2)\big)+\mu(N^2))\right)$
3	$O\left(N\mu(N^2)\log(N)\right)$	$O\left(N(\log(N)+\sqrt{\mu(N^2)})\right)$

Table B.24: Query complexities in our particular problem

can be translated as the condition $\mu(N^2) = O(1)$ (since we do not count the N pairs of a repeated particle), and so the number of oracle calls, in both the worst and average cases, is simply $O(N)$ for the first two algorithms (observe that $\mu(N^2) = O(1)$ allows $B(N^2)$ to be taken as $O(1)$) and $O(N\log(N))$ for the third one. In this situation it seems reasonable to expect that the three algorithms might give accurate outputs even for small values of N.

On the other hand, we might simply assume that $\mu(N^2) = O(N)$ (because of the uniform bound on the number of closed particles to a fixed one), and so the algorithms require queries of the orders given in Table B.25. Notice that, in this case, algorithm 2 (taking the natural choice $B(N^2) = O(N)$) should be avoided, and one can choose between algorithm 1 (in a conservative setting, and if the exact value of $\mu(N^2)$ is known) and algorithm 3 (if only the average running time is of interest).

Algorithm	Worst case	Average case	$\mu(N^2), B(N^2)$
1	$O(N)$	$O(N)$	
2	$O(N)$	$O(N)$	$O(1)$
3	$O(N\log(N)$	$O(N\log N)$	
1	$O\left(N\sqrt{N}\log(N)\right)$	$O\left(N\sqrt{N}\log(N)\right)$	
2	$O\left(N^2\log(N)\right)$	$O\left(N^2\right)$	$O(N)$
3	$O\left(N^2\log(N)\right)$	$O\left(N\sqrt{N}\right)$	

Table B.25: Query complexities when $\mu(N^2), B(N^2) = O(1)$, or $\mu(N^2), B(N^2) = O(N)$

Second step: fix one particle and look for the close ones

Here we have $\nu = N$ and $\mu(N) \leq 27$. If we want to apply the general setting, the requirement on the minimum number of particles is the same as above ($N \geq 54$ for the first algorithm and $N \geq 36$ for the second and third ones). Also, for the first method, we need to assume that $\mu(N)$ has a limit, as $N \to \infty$. Again, this assumption is realistic, since the more particles we have, the more chances of having close particles to a given one, i.e., it seems realistic assuming that $\mu(N)$ is non-decreasing, and so it has a limit. Moreover, in this situation $\mu(N) = O(1)$ always. The need of a knowledge of $\mu(N)$ is, as above, the main obstacle for using the first algorithm.

Application of the general setting yields an asymptotic number of oracle calls that is $O\left(\sqrt{N}\right)$ for the first two methods, and $O(\sqrt{N}\log(N))$ for the third one. This number of oracle queries has to be multiplied by the number of "updated" particles, that we will call $\alpha(N)$. There is still another missing factor that must be taken into account. We know that any of the algorithms provides a uniform success probability $0 < 1 - w < 1$. When we repeat the algorithm $\alpha(N)$ times, the lower bound on the success probability becomes $(1-w)^{\alpha(N)}$, which tends to 0, as $\alpha(N)$ tends to infinity. To avoid this, we can repeat the search method S times for each updated particle, so that the probability that we do not find all the close pairs is bounded from above by $\sum_{i=1}^{\alpha(N)} P(\text{fail to find the neighbour list of the i-th particle in all the S repetitions}) = \alpha(N)w^S$. Then, if we take $S = \left\lceil \frac{\log\left(\frac{\epsilon}{\alpha(N)}\right)}{\log(w)} \right\rceil$, which is $O(\log(\alpha(N)))$, we can make the failure probability less than any given ϵ, in particular w. Therefore, the total amount of oracle calls that we need to consider is $O\left(\sqrt{N}\alpha(N)\log(\alpha(N))\right)$ for the first two algorithms and $O\left(\sqrt{N}\log(N)\alpha(N)\log(\alpha(N))\right)$ for the third one.

Backtracking

A final question to be addressed is when it would be desirable to retake the first approach instead of updating with the second approach. This would happen, for instance, when the number of updated particles, $\alpha(N)$, verifies $\alpha(N)\log(\alpha(N)) \geq N$, but the constants hidden by the O notation can make it interesting even for smaller $\alpha(N)$.

SUPPLEMENTARY MATERIAL

Results for the Algorithm 2 and Algorithm 3 experiments

In this Appendix, we present the full set of results for the experiments performed with Algorithms 2 and 3. In all the cases, we have consider values of R ranging from 5 to 70, number of particles $125, 216, 512$ and 1000, and 550
$\mu = 40, 80, 150$. The results are shown in Tables C.26 through C.49. In all the cases, we present the values of R and the number of times that not all particle pairs were recovered ("Fails"), together with minimum, maximum, average and standard deviation of the number of oracle queries.

R	Fails	Mininum	Maximum	Average	Standard deviation
5	718450	40	7418	2939.6492	1717.0896
10	41249	197	9648	5569.0876	852.5619
15	1505	654	9932	6009.2337	671.0525
20	58	1211	10080	6329.9406	669.5504
25	4	4001	10578	6648.2113	674.7998
30	0	4275	11155	6966.1094	679.7743
35	0	4324	11405	7283.0392	684.7065
40	0	4928	11613	7600.7908	690.3017
45	0	5189	12223	7918.7430	695.0099
50	0	5315	12237	8237.5179	699.4031
55	0	5641	12459	8553.4562	704.2803
60	0	5777	12843	8872.4479	708.3241
65	0	6195	12964	9189.5284	712.8600
70	0	6528	13612	9505.8860	718.6782

Table C.26: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 125 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	2824	386	3712	2228.2688	217.8424
10	67	1165	4018	2548.1559	232.4877
15	1	1813	4630	2866.3376	246.8998
20	0	2027	4928	3183.3695	260.2819
25	0	2322	5271	3500.6617	273.3451
30	0	2597	5657	3818.5138	285.3814
35	0	2701	5959	4136.2548	297.1029
40	0	2964	6127	4453.5569	308.5054
45	0	3190	6810	4770.9495	319.0550
50	0	3463	6904	5088.6793	329.8067
55	0	3823	7400	5405.9980	339.7549
60	0	4015	7586	5723.9451	349.8596
65	0	4329	7833	6040.8812	359.2226
70	0	4647	8415	6358.7057	368.9050

Table C.27: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 125 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	718628	37	15516	5901.4243	3448.7590
10	41357	503	18982	11181.1341	1714.4627
15	1556	1420	20896	12065.4905	1345.8739
20	54	2051	21332	12711.5539	1341.5124
25	2	6818	21172	13349.2033	1353.0338
30	0	8783	22207	13987.1952	1364.4461
35	0	8966	23816	14624.5805	1373.5282
40	0	9563	22670	15264.0114	1382.1603
45	0	10208	23718	15898.3035	1392.8924
50	0	11040	24741	16537.0394	1403.1162
55	0	11684	25428	17174.2995	1412.9613
60	0	12096	26344	17812.5433	1421.5418
65	0	12830	27214	18450.4361	1431.4450
70	0	13201	28141	19089.2051	1441.6392

Table C.28: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 216 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	2377	898	7689	4825.1556	445.2583
10	59	2104	8534	5466.6574	473.8216
15	3	3780	9491	6103.8121	501.5960
20	0	4255	10959	6742.7040	528.9813
25	0	4824	11518	7379.4405	553.3524
30	0	5463	11685	8016.7392	577.5888
35	0	5995	12124	8654.2270	601.4419
40	0	6501	13166	9292.0317	623.7913
45	0	6944	13961	9930.4703	645.1531
50	0	7375	14465	10567.1632	666.4657
55	0	8018	15225	11204.3752	685.5598
60	0	8800	15461	11840.9150	705.6348
65	0	9069	16568	12478.6143	724.6778
70	0	9830	17082	13115.8680	743.3970

Table C.29: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 216 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	718791	160	30868	11825.2023	6914.3259
10	41221	878	38631	22403.9427	3427.6403
15	1517	2051	40597	24178.9512	2702.7087
20	57	4604	39848	25475.3335	2691.4411
25	1	7201	43896	26750.3906	2707.0025
30	0	17789	43981	28031.4853	2729.6234
35	0	18180	44091	29304.0688	2750.3152
40	0	19496	46887	30585.1129	2770.6149
45	0	20388	48264	31859.0301	2788.2287
50	0	21896	49474	33138.2467	2807.0429
55	0	23460	49567	34416.2805	2826.3657
60	0	24140	50592	35691.7640	2850.4040
65	0	24539	51950	36970.7488	2867.3966
70	0	26362	54432	38242.6094	2887.9825

Table C.30: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 512 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	1923	1910	15753	10145.7909	902.9648
10	44	4633	17451	11432.0083	960.5882
15	3	8536	20316	12710.2747	1015.7321
20	0	8806	22982	13986.8836	1067.2772
25	0	10351	22902	15264.6188	1117.9356
30	0	11131	23273	16540.8108	1166.4377
35	0	12455	25087	17819.0396	1212.3312
40	0	13582	26600	19095.3893	1255.3162
45	0	14479	28519	20374.3377	1298.8724
50	0	14841	28812	21648.2902	1342.0320
55	0	16772	30478	22927.2429	1380.5863
60	0	17651	32652	24205.4284	1419.4951
65	0	18730	34298	25486.3516	1457.7054
70	0	19923	34743	26760.8443	1494.1648

Table C.31: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 512 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	718391	166	63409	23683.5436	13833.8447
10	40895	2324	75457	44862.0495	6874.5329
15	1492	5649	77415	48402.0877	5402.9775
20	72	9992	85992	50994.6734	5381.6955
25	2	20000	84930	53551.7566	5423.7290
30	0	34156	90053	56105.2792	5462.8954
35	0	36937	89677	58674.4402	5503.6964
40	0	38349	91133	61227.9910	5542.4209
45	0	40524	97136	63783.1961	5582.8703
50	0	43766	99471	66336.5767	5621.0076
55	0	46921	101023	68902.9617	5656.8642
60	0	46568	104222	71462.3188	5697.8280
65	0	48738	107966	74008.3424	5741.4485
70	0	54323	109460	76582.4883	5770.1841

Table C.32: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 1000 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	1635	4265	31542	20964.4723	1823.0690
10	39	13441	35629	23536.2271	1936.5643
15	0	16590	37738	26093.3338	2046.9211
20	0	19203	43485	28652.9518	2151.9537
25	0	21504	46209	31208.2892	2250.5519
30	0	23149	48898	33770.6442	2342.3008
35	0	24765	49575	36319.7148	2434.8336
40	0	27083	56501	38880.8974	2525.8832
45	0	30029	56789	41439.2587	2609.3545
50	0	31775	58314	43996.7575	2693.1135
55	0	33189	61831	46553.6430	2770.0797
60	0	35264	65534	49114.7834	2850.8926
65	0	38689	67909	51672.2018	2926.7659
70	0	39434	71960	54225.1105	2999.7123

Table C.33: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 1000 particles and $\mu = 40$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	923296	25	13624	3635.9636	2856.8485
10	80365	279	15537	10336.4948	1842.9808
15	2888	624	16060	11098.5778	983.8430
20	98	1196	16695	11430.6496	942.7164
25	5	3176	17785	11750.7350	945.5938
30	0	8209	17179	12066.4249	948.4360
35	0	8317	17552	12385.7663	952.5189
40	0	8636	17778	12701.6750	956.0423
45	0	8748	17962	13021.0638	958.9156
50	0	9394	18442	13336.8252	963.1836
55	0	9767	19120	13654.1344	967.2429
60	0	9806	19565	13972.2885	970.1708
65	0	10307	19695	14291.1269	972.7903
70	0	10502	19788	14607.5681	976.9584

Table C.34: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 125 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	2862	452	4268	2859.8728	231.2177
10	70	1063	4739	3179.8891	244.7497
15	4	2322	5091	3497.5590	258.2530
20	0	2572	5504	3815.2131	271.0638
25	0	2887	5860	4132.6062	283.2911
30	0	3173	6244	4449.8254	295.0987
35	0	3438	6949	4767.5163	306.8322
40	0	3646	6940	5085.1520	317.8033
45	0	3916	7512	5402.9935	327.4926
50	0	4177	7580	5720.4103	338.1065
55	0	4443	7898	6037.9105	348.5818
60	0	4525	8792	6355.0305	357.9535
65	0	5042	8803	6672.3907	367.2009
70	0	5055	9106	6990.2224	376.5325

Table C.35: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 125 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	923535	53	26838	7278.0833	5732.9068
10	80164	466	30978	20764.6058	3686.2588
15	2893	1269	33464	22288.0032	1971.9172
20	98	3378	34485	22956.6472	1888.5995
25	4	6163	33952	23589.5022	1897.7596
30	0	16616	35536	24232.5031	1905.1978
35	0	16131	35181	24866.5963	1909.3065
40	0	18018	36800	25505.5612	1916.9934
45	0	17745	37909	26141.1385	1921.6054
50	0	18943	37610	26782.8205	1929.7497
55	0	19437	37705	27416.0201	1936.2964
60	0	20150	38316	28058.5007	1945.9851
65	0	20475	39788	28694.4024	1955.4758
70	0	20119	40084	29329.3894	1956.2711

Table C.36: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 216 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	2392	1059	9052	6325.5507	474.2337
10	72	2573	10293	6967.4328	500.9015
15	0	5310	11499	7605.0946	527.7988
20	0	5832	11881	8242.9234	552.6728
25	0	6322	13802	8880.8130	577.3092
30	0	6792	13112	9517.5153	600.0381
35	0	7086	14341	10155.0858	621.8501
40	0	7719	14952	10793.2809	644.4091
45	0	8440	15185	11430.6318	664.4279
50	0	8552	15962	12067.8135	685.1698
55	0	9246	16926	12704.4590	703.8530
60	0	9872	17237	13343.4304	723.7423
65	0	10533	17705	13979.7554	743.0180
70	0	10967	18852	14618.0132	760.9026

Table C.37: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 216 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	923253	187	52725	14611.2056	11490.1537
10	80134	1061	63761	41596.0359	7411.9602
15	2842	2385	67161	44658.7835	3943.5957
20	108	3890	67044	46001.3135	3790.5698
25	2	10629	68236	47280.9809	3798.1814
30	0	33531	69521	48549.5045	3805.9196
35	0	32346	70270	49839.0316	3822.6188
40	0	34265	71939	51111.6997	3840.7633
45	0	36299	73514	52389.4730	3858.2295
50	0	37650	74989	53670.5450	3874.8880
55	0	39547	76226	54943.0548	3881.4406
60	0	39633	76825	56222.5949	3896.9765
65	0	41269	79521	57496.3640	3908.8121
70	0	42185	80082	58775.9391	3928.2244

Table C.38: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 512 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	1950	2115	19522	13470.1575	961.4043
10	54	4566	20804	14756.1635	1014.3651
15	4	11101	22472	16033.6005	1066.0397
20	0	12368	24762	17312.6776	1117.6257
25	0	12699	26817	18589.4395	1166.2992
30	0	14627	28121	19865.9294	1212.8327
35	0	15245	28624	21143.3044	1254.8767
40	0	16422	29459	22422.5002	1298.9550
45	0	17223	31994	23698.2628	1339.9998
50	0	18717	32834	24978.4924	1380.1082
55	0	19943	33593	26254.1776	1418.1368
60	0	20652	35096	27534.4299	1456.9480
65	0	21862	38463	28808.9218	1493.8088
70	0	23100	38009	30084.4850	1529.7581

Table C.39: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 512 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	923323	255	104888	29222.7712	2856.8485
10	79935	2390	129834	83289.8695	1842.9808
15	2916	4740	134464	89407.7511	983.8430
20	88	7803	133875	92074.9924	942.7164
25	2	64634	135505	94648.8913	945.5938
30	0	66544	139891	97211.9290	948.4360
35	0	67639	142266	99766.2836	952.5189
40	0	71193	146028	102337.1896	956.0423
45	0	74491	149917	104881.7774	958.9156
50	0	73612	148087	107438.8263	963.1836
55	0	78205	158620	109989.6281	967.2429
60	0	80210	154721	112540.4592	970.1708
65	0	80626	158723	115106.1941	972.7903
70	0	82811	160788	117668.0933	976.9584

Table C.40: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 1000 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	1738	4574	40682	28031.4885	1945.7548
10	56	11198	43337	30607.4803	2048.0667
15	1	19392	46624	33163.4896	2152.5102
20	0	25475	50528	35718.5258	2251.9494
25	0	27591	54305	38278.1468	2346.4738
30	0	30115	56163	40836.8827	2441.6654
35	0	31025	57841	43394.7480	2526.3774
40	0	33172	61129	45955.3550	2610.1783
45	0	36524	62045	48514.9762	2694.3705
50	0	37159	67122	51068.2165	2774.7045
55	0	39510	69679	53620.3119	2851.4596
60	0	43035	73106	56183.0602	2925.0693
65	0	44113	75208	58741.6703	3003.4321
70	0	46962	76631	61300.5532	3071.1791

Table C.41: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 1000 particles and $\mu = 80$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	991625	23	22017	3936.0119	3539.2533
10	141155	229	26817	18238.5071	4088.6913
15	5007	694	27105	19945.3744	1482.5957
20	178	1326	27614	20312.2466	1286.8417
25	7	3865	27146	20631.0459	1280.5572
30	1	15250	27272	20950.8806	1286.0155
35	0	15946	28345	21269.7710	1288.2153
40	0	15910	28047	21584.3052	1291.9420
45	0	16573	29365	21901.8913	1293.7654
50	0	16521	29823	22219.7388	1296.2946
55	0	17168	29288	22536.5065	1297.8384
60	0	17380	29745	22856.1961	1301.6468
65	0	17835	29883	23171.8584	1303.7534
70	0	17672	30728	23490.2205	1306.3670

Table C.42: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 125 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	2954	342	5015	3567.7794	241.7000
10	77	1533	5699	3887.7511	254.1947
15	2	2880	5828	4205.6119	267.2645
20	0	3178	6341	4522.7434	280.1169
25	0	3603	6610	4840.4259	292.1643
30	0	3749	7134	5157.9179	303.3228
35	0	3966	7528	5475.1769	314.3134
40	0	4304	7636	5792.8299	324.8783
45	0	4590	8200	6110.0810	335.2959
50	0	4791	8346	6427.4647	345.2632
55	0	5012	8680	6745.1640	354.6144
60	0	5358	9359	7062.6664	364.8543
65	0	5557	9307	7380.1786	373.4239
70	0	5895	9825	7697.6120	382.8709

Table C.43: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 125 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	991606	56	44866	7900.6697	7117.2576
10	140843	609	53792	36621.1976	8220.3127
15	5104	1370	54612	40048.2808	2980.3527
20	181	2722	54079	40784.4320	2579.0162
25	6	18818	54880	41431.0001	2574.3050
30	0	31161	55336	42063.5458	2579.5387
35	0	31338	56721	42704.7072	2586.1462
40	0	32303	57387	43336.8190	2587.6622
45	0	33082	57541	43977.8210	2591.9175
50	0	33625	58658	44618.2815	2599.5720
55	0	34127	59394	45255.0186	2604.8972
60	0	34587	60574	45886.6671	2610.6267
65	0	34515	59675	46530.9800	2616.4212
70	0	35837	61274	47168.8694	2618.2031

Table C.44: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 216 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	2329	937	11030	8095.4643	497.3407
10	58	5365	12306	8737.4271	522.6686
15	1	6995	12657	9375.3278	548.0367
20	0	7495	13782	10012.7649	572.8338
25	0	7965	14121	10650.1500	596.2434
30	0	8176	15215	11287.6006	618.6946
35	0	9106	16204	11925.8409	640.3208
40	0	9565	16007	12562.8546	660.9694
45	0	10093	17563	13199.8430	681.4702
50	0	10632	17767	13837.7113	700.9023
55	0	11348	18398	14475.9172	720.4413
60	0	11748	19119	15114.0568	738.7913
65	0	12333	20315	15748.6789	757.0198
70	0	12957	20667	16386.4846	774.5531

Table C.45: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 216 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	991722	171	87389	15787.1779	14214.6269
10	141218	1133	104564	73370.9227	16477.8624
15	5119	2584	106443	80267.4885	5956.5470
20	191	8396	110829	81734.6673	5176.2931
25	5	18438	112943	83029.7071	5154.9605
30	0	61565	112791	84295.9919	5163.6106
35	0	63555	112327	85583.6788	5176.9628
40	0	64111	113811	86853.8841	5194.6757
45	0	66188	115603	88128.6841	5194.7385
50	0	66729	117867	89402.4787	5205.8579
55	0	67905	117424	90693.6458	5219.0534
60	0	69315	119052	91965.4294	5231.4581
65	0	69658	124526	93241.4550	5240.3562
70	0	72515	124582	94512.2240	5245.4772

Table C.46: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 512 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	1875	3230	23859	17501.6378	1012.9452
10	54	8240	25333	18787.8901	1061.4635
15	2	15497	26884	20064.1490	1112.5338
20	0	15898	28518	21342.7490	1160.4791
25	0	17546	30946	22622.6177	1206.2732
30	0	18106	32741	23897.8824	1249.8472
35	0	19230	32645	25176.4913	1293.0121
40	0	20251	34749	26454.0754	1335.0615
45	0	21517	35411	27730.7457	1376.7511
50	0	22288	37427	29007.4129	1415.1234
55	0	24033	38662	30284.7649	1452.5908
60	0	24633	39454	31562.9967	1489.6124
65	0	26049	42284	32839.0674	1528.9651
70	0	26944	42412	34118.9010	1563.3344

Table C.47: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 512 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	991505	414	180395	31684.0844	28523.4480
10	140787	2180	206019	146916.3709	32947.0996
15	5019	5198	219209	160687.7883	11937.5891
20	147	12020	221281	163628.5918	10357.0073
25	6	36939	227243	166217.6099	10331.6578
30	0	117389	227737	168767.3977	10334.1867
35	0	127876	226940	171312.8985	10360.0652
40	0	129330	228330	173867.6444	10383.3829
45	0	132929	237269	176427.3668	10392.6679
50	0	135468	235256	178971.1799	10419.4715
55	0	139009	233572	181569.7748	10439.0855
60	0	141583	238208	184098.0041	10464.7435
65	0	142251	242487	186666.0778	10483.9050
70	0	142234	245575	189222.6928	10520.2681

Table C.48: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 2 with 1000 particles and $\mu = 150$

R	Fails	Mininum	Maximum	Average	Standard deviation
5	1673	6193	48431	36746.5779	2044.8347
10	42	30478	53071	39321.0275	2147.6401
15	3	31524	55663	41873.6086	2244.5502
20	0	32971	57502	44433.0812	2337.9566
25	0	35685	62610	46991.5814	2431.9973
30	0	37507	63685	49547.2117	2522.8804
35	0	40071	66778	52107.0978	2607.5169
40	0	40262	69429	54662.2551	2687.0371
45	0	44408	74472	57220.2541	2768.7009
50	0	46548	74999	59780.8423	2842.9850
55	0	48635	78220	62337.2892	2925.5149
60	0	48742	84650	64890.8063	2998.5694
65	0	53761	82784	67451.0666	3066.7362
70	0	55717	86997	70007.5573	3139.5400

Table C.49: # of fails and minimum, maximum, average and standard deviation of the number of oracle queries for 10^6 repetitions of Algorithm 3 with 1000 particles and $\mu = 150$

References

[1] N. March, M. Tosi, Atomic Dynamics in Liquids, Dover Publications, Inc. New York, 1991.

[2] M. Allen, D. Tildesley, Computer Simulation of Liquids, Clarendon Press Oxford, 1989.

[3] B. Hayes, The 100-billion-body problem, American Scientist 103 (90).

[4] J. B. Caballero, A. M. Puertas, A. Fernández-Barbero, F. Javier de las Nieves, Formation of clusters in a mixture of spherical colloidal particles oppositely charged, Colloids and Surfaces A: Physicochemical and Engineering Aspects 270-271 (2005) 285 – 290, liquids and MesoScience.

[5] J. Barnes, P. Hut, A hierarchical o(n log n) force-calculation algorithm, Nature (324) (1986) 446–449.

[6] W. Dehnen, A hierarchical o(n) force calculation algorithm, Journal of Computational Physics 179 (1) (2002) 27 – 42.

[7] A. A. Chialvo, P. G. Debenedetti, On the use of the verlet neighbor list in molecular dynamics, Computer Physics Communications 60 (2) (1990) 215 – 224. doi:https://doi.org/10.1016/0010-4655(90)90007-N.

[8] R. Potestio, C. Peter, K. Kremer, Computer simulations of soft matter: Linking the scales, Entropy 16 (2014) 4199–4245. doi:10.3390/e16084199.

[9] M. A. Nielsen, I. Chuang, Quantum computation and quantum information (2002).

[10] B. Paredes, F. Verstraete, J. I. Cirac, Exploiting quantum parallelism to simulate quantum random many-body systems, Physical review letters 95 (14) (2005) 140501.

[11] D. A. Lidar, A. T. Rezakhani, A. Hamma, Adiabatic approximation with exponential accuracy for many-body systems and quantum computation, Journal of Mathematical Physics 50 (10) (2009) 102106.

[12] A. S. Sørensen, E. Altman, M. Gullans, J. Porto, M. D. Lukin, E. Demler, Adiabatic preparation of many-body states in optical lattices, Physical Review A 81 (6) (2010) 061603.

[13] L. K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 1996, pp. 212–219.

[14] R. Isaac, The pleasures of probability, Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1995, readings in Mathematics. `doi:10.1007/978-1-4612-0819-8`.
URL `https://doi.org/10.1007/978-1-4612-0819-8`

[15] S. Arunachalam, A. Belovs, A. M. Childs, R. Kothari, A. Rosmanis, R. de Wolf, Quantum Coupon Collector, in: S. T. Flammia (Ed.), 15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020), Vol. 158 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2020, pp. 10:1–10:17. `doi:10.4230/LIPIcs.TQC.2020.10`.
URL `https://drops.dagstuhl.de/opus/volltexte/2020/12069`

[16] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, J. I. Latorre, Data reuploading for a universal quantum classifier, Quantum 4 (2020) 226.

[17] M. Mohammadi, M. Eshghi, On figures of merit in reversible and quantum logic designs, Quantum Information Processing 8 (4) (2009) 297–318.

[18] H. Thapliyal, N. Ranganathan, R. Ferreira, Design of a comparator tree based on reversible logic, in: 10th IEEE International Conference on Nanotechnology, IEEE, 2010, pp. 1113–1116.

[19] A review on reversible quantum adders, Journal of Network and Computer Applications 170 (2020) 102810. doi:https://doi.org/10.1016/j.jnca.2020.102810.

[20] H. Thapliyal, N. Ranganathan, Design of efficient reversible logic-based binary and bcd adder circuits, ACM Journal on Emerging Technologies in Computing Systems (JETC) 9 (3) (2013) 17.

[21] A. Nagamani, C. Ramesh, V. K. Agrawal, Design of optimized reversible squaring and sum-of-squares units, Circuits, Systems, and Signal Processing 37 (4) (2018) 1753–1776.

[22] H. Xia, H. Li, H. Zhang, Y. Liang, J. Xin, Novel multi-bit quantum comparators and their application in image binarization, Quantum Information Processing 18 (7) (2019) 229.

[23] H. Thapliyal, Mapping of subtractor and adder-subtractor circuits on reversible quantum gates, in: Transactions on Computational Science XXVII, Springer, 2016, pp. 10–34.

[24] F. Orts, G. Ortega, E. M. Garzón, A faster half subtractor circuit using reversible quantum gates, Baltic Journal of Modern Computing 7 (1) (2019) 99–111.

[25] H. Thapliyal, H. Jayashree, A. Nagamani, H. R. Arabnia, Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder, in: Transactions on Computational Science XVII, Springer, 2013, pp. 73–97.

[26] T. G. Draper, S. A. Kutin, E. M. Rains, K. M. Svore, A logarithmic-depth quantum carry-lookahead adder, arXiv preprint quant-ph/0406142.

[27] H. Bhagyalakshmi, M. Venkatesha, Optimized multiplier using reversible multi-control input toffoli gates, International Journal of VLSI Design & Communication Systems 3 (6) (2012) 27.

[28] H. Rangaraju, A. B. Suresh, K. Muralidhara, Design and optimization of reversible multiplier circuit, International Journal of Computer Applications 52 (10).

[29] M. S. Islam, M. Rahman, Z. Begum, M. Z. Hafiz, Low cost quantum realization of reversible multiplier circuit, Information technology journal 8 (2) (2009) 208–213.

[30] H. Bhagyalakshmi, M. Venkatesha, An improved design of a multiplier using reversible logic gates, International journal of engineering science and technology 2 (8) (2010) 3838–3845.

[31] D. Wang, Z.-H. Liu, W.-N. Zhu, S.-Z. Li, Design of quantum comparator based on extended general toffoli gates with multiple targets, Computer Science 39 (9) (2012) 302–306.

[32] A. N. Al-Rabadi, Closed-system quantum logic network implementation of the viterbi algorithm, Facta universitatis-series: Electronics and Energetics 22 (1) (2009) 1–33.

[33] C. Vudadha, P. S. Phaneendra, V. Sreehari, S. E. Ahmed, N. M. Muthukrishnan, M. B. Srinivas, Design of prefix-based optimal reversible comparator, in: 2012 IEEE Computer Society Annual Symposium on VLSI, IEEE, 2012, pp. 201–206.

[34] H. Xia, H. Li, H. Zhang, Y. Liang, J. Xin, An efficient design of reversible multi-bit quantum comparator via only a single ancillary bit, International Journal of Theoretical Physics 57 (12) (2018) 3727–3744.

[35] M. Boyer, G. Brassard, P. Høyer, A. Tapp, Tight bounds on quantum searching, Fortschr. Phys 46 (4-5) (1998) 493–505.

2.3 MultiDimensional Scaling

One publication has been published as part of our efforts in this area:

F. Orts, G. Ortega, E.M. Garzón and A.M. Puertas. Improving the energy efficiency of SMACOF for multidimensional scaling on modern architectures. *Journal of Supercomputing*, 75, 1038-1050, 2018. JCR (2018) = 2.157. Subject categories = Computer Science, Theory & Methods: 35/105 (**Q2**); Computer Science, Hardware & Architecture: 22/53 (Q2); Engineering, Electrical & Electronic: 132/266 (Q2).

2.3.1 Improving the energy efficiency of SMACOF for multidimensional scaling on modern architectures

Title	Improving the energy efficiency of SMACOF for multidimensional scaling on modern architectures		
Authors	*F. Orts, E. Filatovas, G. Ortega, O. Kurasova and E.M. Garzón*		
Journal	Journal of Supercomputing		
Year	2018		
Volume	75		
Pages	1038-1050		
DOI	https://doi.org/10.1007/s11227-018-2285-x		
IF (JCR 2018)	2.157		
Categories	Computer Science, Hardware & Architecture:	22/53	(Q2)
	Computer Science, Theory & Methods:	35/105	(Q2)
	Engineering, Electrical & Electronic	132/266	(Q2)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

J Supercomput (2019) 75:1038–1050
https://doi.org/10.1007/s11227-018-2285-x

Improving the energy efficiency of SMACOF for multidimensional scaling on modern architectures

F. Orts[1] · E. Filatovas[2] · G. Ortega[1] · O. Kurasova[3] · E. M. Garzón[1]

Published online: 27 February 2018

Abstract The reduction of the dimensionality is of great interest in the context of big data processing. Multidimensional scaling methods (MDS) are techniques for dimensionality reduction, where data from a high-dimensional space are mapped into a lower-dimensional space. Such methods consume relevant computational resources; therefore, intensive research has been developed to accelerate them. In this work, two efficient parallel versions of the well-known and precise SMACOF algorithm to

This work has been partially supported by the Spanish Ministry of Science throughout Projects TIN2015-66680 and CAPAP-H5 network TIN2014-53522, by J. Andalucía through Projects P12-TIC-301 and P11-TIC7176, by the European Regional Development Fund (ERDF), and by the European COST Action IC1305: Network for sustainable Ultrascale computing (NESUS).

G. Ortega
gloriaortega@ual.es

F. Orts
francisco.orts@ual.es

E. Filatovas
ernest.filatov@gmail.com

O. Kurasova
olga.kurasova@mii.vu.lt

E. M. Garzón
gmartin@ual.es

1 Group of Supercomputation-Algorithms, Department of Informatics, University of Almería, ceiA3, 04120 Almería, Spain

2 Faculty of Fundamental Science, Vilnius Gediminas Technical University, Saulėtekio avn. 11, 10223 Vilnius, Lithuania

3 Institute of Data Science and Digital Technologies, Vilnius University, Akademijos str. 4, 08663 Vilnius, Lithuania

solve MDS problems have been developed and evaluated on multicore and GPU. To help the user of SMACOF, we provide these parallel versions and a complementary Python code based on a heuristic approach to explore the optimal configuration of the parallel SMACOF algorithm on the available platforms in terms of energy efficiency (GFLOPs/watt). Three platforms, 64 and 12 CPU-cores and a GPU device, have been considered for the experimental evaluation.

Keywords Dimensionality reduction · Multidimensional scaling · Energy efficiency · SMACOF algorithm

1 Introduction

Real-world data, such as speech signals, images, biomedical, financial, telecommunication and other data usually have a high dimensionality as each data instance (point) is characterized by a set of features. The dimensionality of such data, as well as the amount of data to be processed, is constantly increasing therefore the requirement of processing these data within a reasonable time frame still remains an open problem. Dimensionality reduction methods which aim to map high-dimensional data into a lower-dimensional space play extremely important role when exploring large datasets. Among such methods multidimensional scaling (MDS) remains one of the most popular [2,8].

One of the dimensionality reduction applications is a graphical visualization of the structure of the high-dimensional data in 2D or 3D space for easier data understanding. Some applications in this line can be found in [12,18,22]. Moreover, MDS has proven to be useful as a technique to evaluate criteria of objects classification [14] or discover criteria which initially had not been taken into account [1], serving as a psychological model that allows to discover human patterns [15].

A well-known algorithm for MDS is called SMACOF (Scaling by Majorizing a COmplicated Function) [7]. The experimental investigation has demonstrated that SMACOF is most accurate algorithm comparing to others [16]. It should be noted that the SMACOF algorithm is the most expensive, as its complexity is $O(m^2)$, where m is the number of observations. Several different approaches have been developed to reduce computational complexity of the MDS techniques. In [23], the complexity was reduced to $O(m\sqrt{m})$ by developing iterative MDS spring model. In [32], authors reduced the complexity to $O(m \log m)$ by dividing the original matrix into sub-matrices and then combining the sub-solutions to obtain a final solution. The improved versions of MDS reduce complexity insignificantly, however, optimization accuracy suffers [16]. Consequently, SMACOF version of MDS is usually chosen as it ensures the sufficient accuracy that is essential in many dimensional cases. In short, the MDS techniques remain in high time complexity order therefore parallel strategies should be considered to accelerate the computation of the MDS procedure [24].

During the last decade, the high-performance computing (HPC) has greatly improved and has been widely applied for MDS techniques. In [29], authors proposed a MDS parallel implementation and explored it under MPI and other libraries. In [11], Fester et al. proposed a CUDA implementation of MDS algorithm based on the high throughput multidimensional scaling (HiT-MDS). In [28], authors suggested a new efficient parallel GPU algorithm for MDS based on virtual particle dynamics [9] and experimentally compared it with multicore CPU version. In [16], the multi-level MDS Glimmer algorithm was developed for GPU by dividing the input data into hierarchical levels and executing the algorithm recursively. It must be noted that currently Glimmer is the most well-known and widely used GPU tool for MDS. Another CUDA-based technique to get MDS approximation is CFMDS [27] that implements both single-level and multilevel approaches.

In [26], authors proposed a correlation clustering framework which uses MDS for layout and GPU-acceleration to speedup visual feedback. In [25], GPU version of MDS was developed to improve content-based image retrieval (CBIR) systems. Summarizing, the research on this HPC field is being carried out actively; it remains relevant as the new GPU architecture and heterogeneous platforms constantly appear, and should be effectively exploited for solving dimensionality reduction problems of different complexities.

Currently, the target of HPC includes the optimization of energy consumption. The ratio of the computational speed to the electrical power (*GFLOPs/watt*) is usually defined as a parameter that is a suitable indicator of the energy efficiency [19]. The increase in this parameter means that the system achieves better performance (GFLOPs) with less electrical power (watts) and, as consequence, less energy is consumed. Therefore, for the optimal parallel executions of SMACOF, the ratio should be maximized.

In this paper, parallel versions of the SMACOF algorithm on multicore and GPU are developed and evaluated on prototypes of modern architectures. As the parallel SMACOF algorithm can be executed on different alternative platforms, the kind of platform and its resources that optimize the runtime and/or energy efficiency need to be determined. Bearing in mind that the parallel performance depends on the problem sizes, the users of parallel SMACOF need support to configure it. For this purpose, a benchmarking process to find the optimal solutions has been developed. It is based on a heuristic approach which combines two concepts: the analysis of the first iterations of SMACOF representative computation and functional models of performance and power consumption of homogeneous parallel platforms. The benchmarking process has been evaluated using different platforms (multicore and GPU) and various sizes of the problem. Moreover, the energy efficiency of SMACOF has been experimentally evaluated on two different multicore platforms and a GPU device.

The paper is organized as follows. In Sect. 2, the descriptions of the Multidimensional Scaling and the SMACOF algorithm are provided. Section 3 describes the proposed multicore and GPU parallel implementations of the SMACOF algorithm. In Sect. 4, the algorithm for tuning the energy efficiency of SMACOF is presented. Experimental evaluations of the parallel implementations on three platforms are discussed in Sect. 5. Finally, conclusions are drawn in Sect. 6.

Algorithm 1 SMACOF($m, s, \Delta, kmax, \epsilon, Y$)

```
Require:
    $m$: number of items;
    $s$: dimension of low-dimensional space;
    $\Delta$: $m \times m$ matrix of dissimilarities of observed data on the high-dimensional space ($n$);
    $kmax$: maximum number of iterations;
    $\epsilon$: threshold for the stress variance
Ensure:
    $Y$: set of finding points in the low-dimensional space stored in a $m \times s$ matrix
1: Initial Solution randomly generated, $Y^0$
2: Compute Euclidean distances, $D^0 = [d(Y_i^0, Y_j^0)]$    ▷ $O(m^2 s)$
3: $k = 0$, $error = 1$
4: if ($k < kmax$) and ($error > \epsilon$) then
5:     Compute Guttman transform matrix, $B^k \equiv B^k(\Delta, D^{k-1})$ (Algorithm 2)    ▷ $O(m^2)$
6:     Compute Guttman transform, $Y^k = 1/m \cdot B^k \cdot Y^{k-1}$    ▷ $O(m^2 s)$
7:     Update distances $D^k = [d(Y_i^k, Y_j^k)]$    ▷ $O(m^2 s)$
8:     Compute $E_{MDS}^k$ (Eq. 1)
9:     $error = |E_{MDS}^k - E_{MDS}^{k-1}|$
10:    $k = k + 1$
11: return $Y$
```

2 SMACOF algorithm for MDS

Multidimensional scaling is a technique for the analysis of similarity or dissimilarity data on a set of objects (items). It aims at finding points $Y_1, Y_2, \ldots, Y_m \equiv Y$ in the low-dimensional space $\mathbb{R}^s$, $s < n$, such that the distances between them are as close as possible to the distances between the original points $X_1, X_2, \ldots, X_m \equiv X$ in the space $\mathbb{R}^n$. This is achieved by minimizing the stress function:

$$E_{\text{MDS}} = \sum_{i<j} \left(\delta_{ij} - d(Y_i, Y_j)\right)^2 \tag{1}$$

Here, $d(\cdot, \cdot)$ (δ) is the distance between two points in the low-dimensional space (multidimensional space).

There are many strategies to solve MDS problems [8]. We focus our attention on the well-known SMACOF algorithm which is based on a particular minimization process of the stress function [7]. The theoretical background of SMACOF is simpler and more powerful than other approaches from convex analysis, because it guarantees monotone convergence of stress [2]. SMACOF has demonstrated better results when optimizing stress function comparing to other proposals in the literature [16]. The main idea is based on the majorizing concept which consists in approximating a complex function by another one simpler. This method iteratively finds a new function, which is located above the original function and touches at the supporting point. At every iteration of the algorithm, the minimum of the new function is closer of the minimum of the complex function, in our case the stress function [2]. SMACOF can be expressed by Algorithm 1 in which the complexity order of the most relevant tasks appeared between parenthesis.

Algorithm 2 Two pseudocodes to compute B^k (line 5 of Alg 1): On the left, the approach from Eq. 8.24 of [2]; on the right, the two nested loops are collapsed in only one to later obtain a balanced parallel execution of Guttman transform matrix

```
Require:
    $m$: number of $items$;
    $\Delta$: $[\delta_{ij}]$, $m \times m$ matrix of dissimilarities;
    $D$: $[d_{ij}]$, Euclidean distances matrix
Ensure:
    $B$: $[b_{ij}]$, Guttman transform matrix
1: for $i = 0; i < m; i++$ do
2:     for $j = i + 1; j < m; j++$ do
3:         if $d_{ij} \neq 0$ then
4:             $b_{ij} = -\delta_{ij}/d_{ij}$
5:         else
6:             $b_{ij} = 0$
7: for $i = 0; i < m; i++$ do
8:     $b_{ii} = -\sum_{j=1, j\neq i}^{m} b_{ij}$
9: return $B$
```

```
Require:
    $m$: number of $items$;
    $\Delta$: $[\delta_{ij}]$, $m \times m$ matrix of dissimilarities;
    $D$: $[d_{ij}]$, Euclidean distances matrix
Ensure:
    $B$: $[b_{ij}]$, Guttman transform matrix
1: for $l = 0; l < (m \cdot (m + 1)/2); l++$ do
2:     $i = \lfloor l/(m + 1)\rfloor$, $j = l\%(m + 1)$
3:     if $j > i$ then
4:         $i = m - i - 1$, $j = m - j$
5:     if $d_{ij} \neq 0$ then
6:         $b_{ij} = -\delta_{ij}/d_{ij}$
7:     else
8:         $b_{ij} = 0$
9:     $b_{ji} = b_{ij}$
10: for $i = 0; i < m; i++$ do
11:     $b_{ii} = -\sum_{j=1, j\neq i}^{m} b_{ij}$
12: return $B$
```

Algorithm 1 has a high computational cost and high memory requirements due to the large data structures involved: input matrix Δ ($m \times m$), output and auxiliary matrices ($m \times s$) and three auxiliary matrices ($m \times m$) to store the similarities among the objects of the low-dimensional space. The symmetry has not been exploited in the storage of the data structures; however, it has been considered for the above-mentioned matrices update. Bearing in mind this fact, the number of floating point operations of Algorithm 1 is: $3s/2m^2 + 3s/2m$ for the initialization (line 2 of Algorithm 1) and $(7/2s + 3/2)m^2 + 1/2(3s + 1)m$ for the iterative process.

3 Parallel implementations of the SMACOF algorithm

The SMACOF computational cost is $O(s \cdot m^2)$ and memory requirements are $O(m^2)$. This feature has limited for years the applicability of SMACOF to solve large MDS problems. The use of HPC techniques helps to overcome this drawback. In this work, we propose two parallel versions based on the exploitation of large-scale modern multicore and GPU architectures. This section is devoted to describing these parallel implementations.

Both implementations are focused on the parallel execution of the computation of the Euclidean distances matrices (lines 2 and 7 of Algorithm 1) and the Guttman transform (lines 5 and 6 of Algorithm 1). Parallel procedures are highlighted in bold in Algorithm 1. To calculate the outputs of these procedures, we have taken into account that we are working with symmetric matrices (B^k, D^k and Δ). For example, to compute the symmetric matrix B^k (which defines Guttman transform) is only necessary to calculate a triangular sub-matrix of $L = (m \cdot (m+1)/2)$ elements. Thus, B^k can be managed as a unidimensional vector of L elements which can be updated in parallel. To distribute this computation among the processing elements, the left part of Algorithm 2 has been transformed into the right one. This way, two nested loops are collapsed into a regular loop to compute the triangular matrix of L elements. It can be easily parallelized with maintaining the load balance. This idea has also been applied to the parallel computation of D^k.

The multicore version has been implemented using C, OpenMP [3] and MKL library [17]. The parallel computations of B^k and D^k consider the symmetry of these matrices. Therefore, Algorithm 2 on the right is taken as reference for the parallel computation of B^k. The l-loop of such algorithm is distributed among the cores and when it has finished a synchronization point is included to ensure that the non-diagonal elements of B^k are computed before starting the parallel i-loop. Moreover, the MKL library (concretely the *cblas_dgemm* routine) is in charge of computing in parallel the matrix-matrix product linked to the Guttman transform (line 6 of the Algorithm 1).

In the GPU version, three kernels have been coded using C and CUDA to compute in parallel D^k (lines 2 and 7 of Algorithm 1) and B^k (line 5 of Algorithm 1). The Euclidean distances require one kernel, and the Guttman transform requires two, as it is explained below. To compute the distances matrix, every thread updates two symmetric elements of D^k matrix. Moreover, shuffle instructions have been used for the reductions involved in the computation of D^k elements. These instructions, available from Kepler NVIDIA architecture, essentially allow threads in the same warp to share information. They can improve the reduction processes [6]. In our experiments, shuffle instructions have demonstrated to improve the performance compared to the reductions based on shared memory. We have observed that the advantage of shuffle instructions versus the shared memory version increases with s. Specifically, we have evaluated the performance for sizes of problem from $m = 10{,}000$ to $m = 40{,}000$ with $s = 64$ and the shuffle version has obtained the same or better performance (up 30%) than shared memory version in the computation of D^k matrix (lines 2 and 7 of Algorithm 1).

The CUDA version of Algorithm 2 to compute B^k on GPU consists of two kernels. In the first kernel, each thread starts by calculating a non-diagonal element of B^k. Next, its symmetric element is copied without requiring any synchronization. When this kernel finishes, the second one computes the diagonal elements from the non-diagonal ones. For Y^k, *cublasDgemm* routine from the cuBLAS library [5] has been used to accelerate matrix-matrix product on GPU (line 6 of Algorithm 1).

4 Tuning the energy efficiency of the SMACOF algorithm

In this work, two parallel implementations have been developed to accelerate the SMACOF algorithm. When solving real-world problems, it is reasonable to run the most energy efficient parallel SMACOF version on a particular subset of resources of available computational platforms.

The idea consists in an initial benchmarking that identifies, for every available platform, the optimal selection of resources for a size of problem of interest. Then, the user can choose the optimal platform for the subsequent execution of the SMACOF algorithm. According to the developed parallel versions, multicore processors and GPUs are considered as target platforms in this work.

The energy efficiency (EE) is usually defined as the ratio of the computational speed to the electrical power, that is *GFLOPs/watt* [19]. Therefore, for the optimal parallel executions of SMACOF, the ratio should be maximized.

The optimization of the EE of parallel applications on modern platforms can be viewed as a problem of scheduling parallel machines with costs [30]. The parallel SMACOF versions can be executed on one of the alternative platforms, for example the different multicore or GPUs architectures. Every platform is denoted by $\mathcal{F}^k \in \mathcal{F}$, $k = 1, \ldots, f$ where $\mathcal{F}$ is the set of f available parallel platforms. Every platform $\mathcal{F}^k$ consists in a set of parallel machines $\mathcal{M}^k$, $\mathcal{F}^k = \{\mathcal{M}_i^k\}_{i=1}^{c_k}$ where c_k is the number of available machines of the platform k. The corresponding energy efficiency depends on the number of machines involved in the computation and the particular input size.

Then, the solution of the scheduling problem corresponds to the subset of platforms $\mathcal{F}^{k_o} \subseteq \mathcal{F}$ with their optimal configurations defined by the machines number $r_{k_o}^o$ that optimizes EE ($r_{k_o}^o \leq c_{k_o}$). We propose a heuristic approach for solving this problem. It is based on a functional model of EE for modern platforms (multicore and GPUs) and the definition of the significant computation in the SMACOF algorithm.

The functional performance models were introduced by Lastovetsky [4,33]. The processor performance depends on the problem size and can be empirically estimated by a benchmarking process. In this way, the modeling performance depends on the combination of the architecture and the application. In similar lines, other authors have been focussed on the benchmarking and they have proposed the concepts of application signature and small-scale executions [10,31]. If the parallel application is iterative, then a subset of iterations can define the significant portion of the application and can be used in the benchmarking [20,21].

The models to estimate EE have to combine performance and power. Previous works have proposed functional models for the EE estimation on iterative applications [13]. If it is focused on a particular execution of the application with F floating point operations on one homogeneous platform k, and it is assumed a perfect load balance among r_k actives machines, then the following model of EE as function of r_k is reasonable:

$$EE(r_k) = \frac{F}{\mathcal{T}^k(r_k)\mathcal{P}^k(r_k)} = \frac{F}{\left(\frac{\mathcal{T}^k(1)}{r_k} + \mathcal{TC}^k(r_k)\right)\left(\mathcal{P}_{idle}^k + r_k p^k(r_k)\right)} \tag{2}$$

where $\mathcal{T}^k(r_k)$ and $\mathcal{P}^k(r_k)$ are the runtime and power consumption on r_k machines respectively, $\mathcal{TC}^k(r_k)$ represents the runtime penalties due to the contention among the actives machines on the k platform, $\mathcal{P}_{idle}^k$ represents the idle power consumption when no process is actively using any machine and $p^k(r_k)$ is the contribution to the power of every machine.

According to this model $\mathcal{T}^k(r_k)$, one minimum for a number of active machines is obtained since $\mathcal{TC}^k(r_k)$ is an increasing function and $\mathcal{P}^k(r_k)$ is also an increasing function for r_k. Therefore, $EE(r_k)$ achieves a maximum for $r_{k_o}^o$ machines.

Then, from the point of view of the SMACOF usage, to optimize EE, it should be identified r_k^o on the set of available platforms for the sizes problem and choose the

Algorithm 3 Heuristic for computing the set of optimal platforms $\{k_o\}$, with their configurations $\{r^o_{k_o}\}$, which optimize the EE of the SMACOF

```
Require:
  $\mathcal{F} = \{\mathcal{F}^k\}_{k=1}^{f}$ with $\mathcal{F}^k = \{\mathcal{M}_i^k\}_{i=1}^{c_k}$;                ▷ Set of platforms
  Parallel versions of SMACOF($m, s, \Delta, kmax, \epsilon, Y$) to execute on the $f$ available platforms;
  $m$ (items), $s$ (output dimensions);                                  ▷ Particular data size
  $sampling$.
Ensure:
  $\{k_o, r^o_{k_o}\}$ optimize the EE on the $f$ available platforms
1: Evaluate the number of FLOAT operations of SMACOF($m, s, \Delta, kmax, \epsilon, Y$)
2: for $k \leftarrow 1$ to $f$ do
3:   Execute Parallel SMACOF($m, s, \Delta, kmax, \epsilon, Y$) on $r_k = c_k$ machines and evaluate its EE denoted
     by $\mathcal{EE}^k$
4:   for $i \leftarrow c_k - sampling$ to $sampling$ do
5:     Execute Parallel SMACOF($m, s, \Delta, kmax, \epsilon, Y$) on $r_k = i$ machines and evaluate $\mathcal{EE}^{Aux}$
6:     if $\mathcal{EE}^{Aux} \leq \mathcal{EE}^k$ then
7:       $r^o_k = i + sampling$
8:       Break $i$-loop
9:     else
10:      $\mathcal{EE}^k = \mathcal{EE}^{Aux}$
11: Select the platforms $\{k_o\}$ with their optimal configurations $\{r^o_{k_o}\}$ which maximize EE
12: return $\{k_o, r^o_{k_o}\}$
```

platform k_o which optimizes EE, i.e. achieves $EE(r^o_{k_o})$. Modern computers provide two different platforms, multicore processors and GPUs and the number of kinds of platforms can increase if clusters of heterogeneous nodes with several kinds of multicore and GPUs platforms are available. We have defined a heuristic to decide what is the best platform to run their particular instances of the parallel SMACOF. Our proposal is organized in two stages, first the identification of the optimal configuration of every platform and second the selection of optimal platforms and configurations. Previous considerations about the EE model help us to define an efficient benchmarking exploration to find the optimal configurations on every platform. Therefore, selective search described in Algorithm 3 can be used to find the optimal platforms and their configurations in the benchmarking process.

As above mentioned, the benchmarking is usually based on the execution of a significant core of the application. SMACOF consists in an iterative procedure to compute the Guttman transforms. The computational cost of every iteration is the same; therefore, a subset of iterations can be considered as the SMACOF significant core to compute the profiling in a efficient way. SMACOF can be configured using the information provided by this preprocess based on exploration of several resources selection on particular combinations of platforms and data sizes.

5 Experimental evaluation

In this section, the SMACOF algorithm to solve MDS problems is evaluated in terms of runtime and energy efficiency on three computational architectures:

$\mathcal{F}_1$: Bullion S8: 4 Intel Xeon E7 8860v3 (16 × 4 CPU-cores);

Table 1 Test problems using several number of items (m), dimensions of multidimensional space (n), and dimensions of low-dimensional space (s)

	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$	$T9$	$T10$	$T11$
m	2000	4000	6000	8000	10,000	12,000	14,000	16,000	18,000	20,000	22,000
n	100	200	300	400	500	600	700	800	900	1000	1100
s	2	3	4	5	6	7	8	9	10	11	12

$\mathcal{F}_2$: Bullx R421-E4 Intel Xeon E5 2620v2 (12 CPU-cores and 64 GB RAM);
$\mathcal{F}_3$: NVIDIA K80 (composed by two Kepler GK210 GPUs) connected to the host Bullx R421-E4 Intel Xeon E5 2620v2.

$\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ run Ubuntu 16.04 LTS and $\mathcal{F}_3$ runs CUDA Toolkit 8. The programs have been compiled using gcc 5.4.0 and nvcc 8.0.44 with optimization flags O3 for GPU architecture 3.5. For the acquisition of energy measurement data, we have collected this information from various hardware counters. For Intel, we have used the Running Average Power Limit (RAPL) interface and, for NVIDIA, the NVIDIA Management Library (NVML).

For the evaluation of SMACOF, test problems of different sizes defined by values of m, n and s have been considered (see Table 1). For this experimental investigation, randomly generated input data were used. The number of evaluated iterations of the SMACOF algorithm has been 100.

Figure 1 shows, the runtime, power and energy efficiency of the set of test problems on $\mathcal{F}_1$ and $\mathcal{F}_2$ (multicore) platforms and Table 2 shows similar parameters on $\mathcal{F}_3$ (GPU) platform with the same test cases. Execution times of the multicore versions (plotted on the top of Fig. 1) are according with runtime models described in Sect. 4. The runtime decreases with the values of r_1 and r_2; therefore, the best performance is achieved for the maximum number of cores. The experimental power measurements are plotted in the middle of Fig. 1. It is remarkable that the temporal evolution of the power partially depends on unpredictable factors for programmers. To overcome this drawback, it has been necessary to collect the measurements after an activity period on the processor to minimize their variance due to changes in the temperature. This instability can be observed in the power plot for both platforms, but we can conclude that power consumption trend increases as the number of cores and the size of the problem.

Focusing our attention on the energy efficiency (plotted on the bottom of Fig. 1), it increases as the number of cores. The highest values of r_1 and r_2 optimize the energy efficiency for the plateau in the plot. Therefore, the optimal value of r_k in both platforms is in a wide interval, for instance 32–64 (10–12) for $\mathcal{F}_1$ ($\mathcal{F}_2$).

To choose the optimal platform, we could compare the three platforms in terms of performance. This way, the best option for $T11$ is $\mathcal{F}_1$, since the execution times are 46.6, 96.2 and 91.5 s on $\mathcal{F}_1$ with $r_1^o = 64$, $\mathcal{F}_2$ with $r_2^o = 12$ and $\mathcal{F}_3$, respectively. This selection is the same for all test cases. If we focus on the energy efficiency, the best option is the GPU when the problem size is enough high since it consumes less power than $\mathcal{F}_1$ and achieves a reasonable performance. Then, to optimize the energy

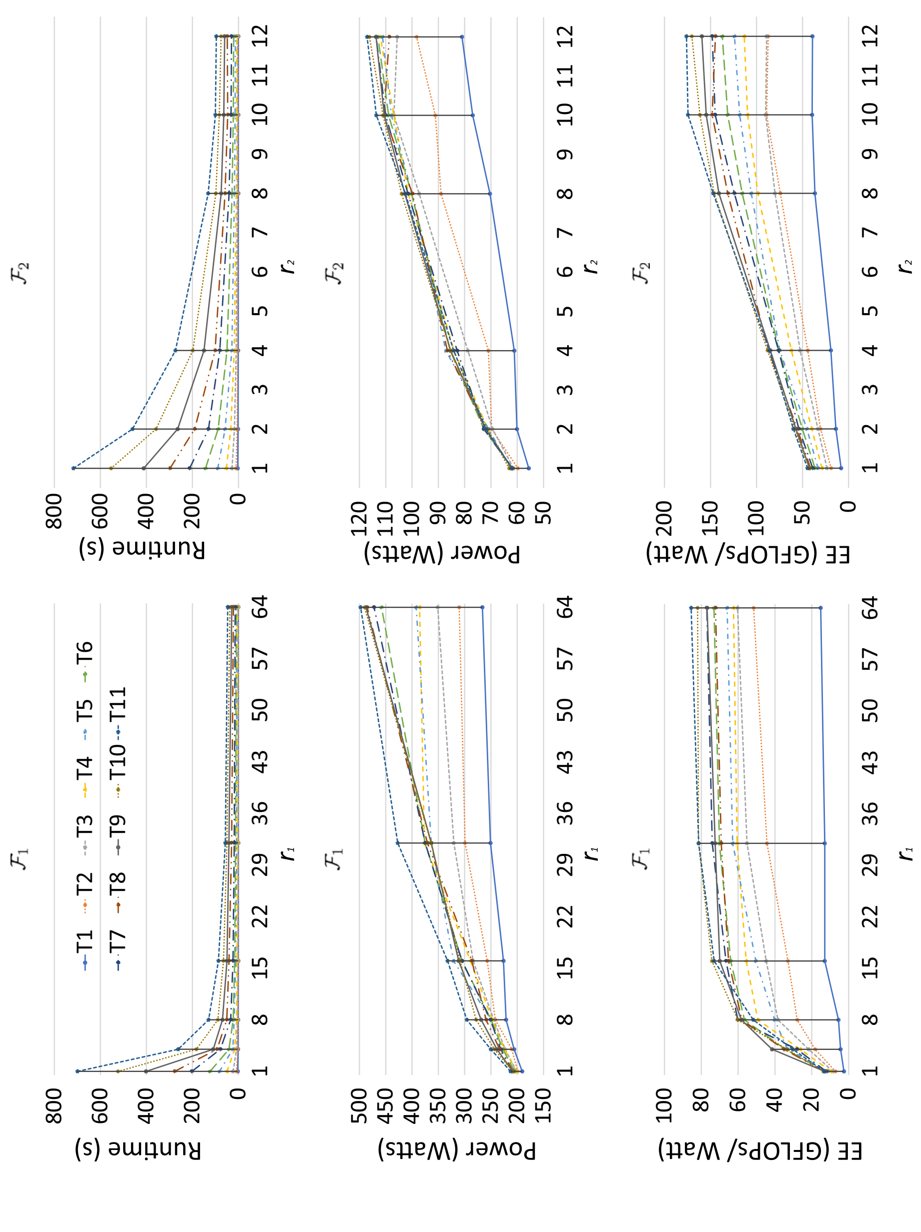


Fig. 1 Runtime (top), power (middle) and energy efficiency (bottom) of the set of test problems (Table 1) on $\mathcal{F}_1$ and $\mathcal{F}_2$ platforms

Table 2 Runtime, power and energy efficiency of the set of test problems (Table 1) on $\mathcal{F}_3$ (GPU) platform

$\mathcal{F}_3$	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$	$T8$	$T9$	$T10$	$T11$
Time (s)	2.8	4.9	5.1	5.9	6.5	11.0	18.8	28.7	42.3	64.9	91.5
Power (W)	38.7	98.6	105.2	108.0	112.6	113.6	112.8	110.1	112.3	111.5	110.3
EE (GFLOPs/W)	13.8	24.7	75.1	150.0	255.1	257.2	240.7	241.7	228.7	205.9	196.6

Table 3 Sampling of EE for $T11$ test according to the benchmarking proposed in Algorithm 3 for multicore platforms $\mathcal{F}_1$ and $\mathcal{F}_2$

$\mathcal{F}_1$			$\mathcal{F}_2$		
r_1	64	61	r_2	12	9
EE (GFLOPs/W)	85.5	85.0	EE (GFLOPs/W)	176.1	155.8

efficiency, the best option is the use of the GPU platform for the test cases $T4 - T11$. For instance for $T11$, the energy efficiencies on the different platforms are 85.5, 176.1 and 196.6 GFLOPs/watt for $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$, respectively. The best platform for $T1-T3$ is the multicore $\mathcal{F}_2$ which consumes less power than $\mathcal{F}_1$.

These results support the benchmarking process explained in Sect. 4 to explore in an automatic way the selection of the optimal parallel platform and its best resource selection. This procedure has been developed in Python. We have chosen sampling $= 3$ to obtain relevant differences between successive experimental evaluations on both platforms. The results support the idea of starting the benchmarking process by the highest numbers of CPU-cores available on every platform to find the optimal r_k is efficient. To illustrate the behavior of the benchmarking (Algorithm 3) for multicore platforms, we focus on the $T11$ test. Table 3 shows the EE obtained when a set of ten iterations of SMACOF are executed on platforms $\mathcal{F}_1$ and $\mathcal{F}_2$. Only two samples for the benchmarking exploration are required for $T11$ since $r_1^o = 64$ and $r_2^o = 12$ are identified by the preprocess. So, we can conclude that the proposed benchmarking can execute an efficient exploration to optimize the energy efficiency of parallel SMACOF.

6 Conclusions

This work has analyzed an approach to optimize the energy efficiency (GFLOPs/watt) of the SMACOF algorithm, a well-known and precise method to solve MDS problems. Two parallel versions of SMACOF, multicore and GPU, have been developed and evaluated. To help the user of SMACOF parallel codes, we provide these versions and a complementary Python code based on a heuristic approach to explore the optimum configuration of the available platforms.

An experimental evaluation has been carried out on three platforms based on architectures with 64CPU-cores, 12CPU-cores and a GPU device. The results show 64-cores processor is the best platform to optimize the runtime of SMACOF; the 12-cores processor is the best option to improve the energy efficiency for the smallest test problems and, for the largest test problems, the optimal energy efficiency is achieved on the GPU.

In currently known parallel versions of SMACOF, only the runtime is considered, and neither the energy consumption nor adaptive capability to the platform and problem size are optimized. Therefore, our SMACOF implementation is of great interest for developing energy efficiency aware applications based on MDS problems. Our implemented versions of the SMACOF algorithm are freely available through the following website: https://github.com/2forts/SMACOF. As future work, we consider to implement a distributed parallel version of SMACOF and to analyze and develop other methods for solving MDS problems.

References

1. Bilsky W, Borg I, Wetzels P (1994) Assessing conflict tactics in close relationships: a reanalysis of a research instrument. In: Hox JJ, Mellenbergh GJ, PG Swanborn (eds.) Facet Theory. Analysis and design, SETOS, Zeist, pp 39–46
2. Borg I, Groenen PJ (2005) Modern multidimensional scaling: theory and applications. Springer, Berlin
3. Chapman B, Jost G, Pas Rvd (2007) Using OpenMP: portable shared memory parallel programming (scientific and engineering computation). The MIT Press, Cambridge
4. Clarke D, Ilic A, Lastovetsky A, Rychkov V, Sousa L, Zhong Z (2014) Design and optimization of scientific applications for highly heterogeneous and hierarchical HPC platforms using functional computation performance models. Wiley, Hoboken, pp 235–260
5. cuBLAS library (2017) http://docs.nvidia.com/cuda/cublas/index.html. Accessed 24 Feb 2018
6. CUDA Pro Tip: Do The Kepler Shuffle (2017) https://devblogs.nvidia.com/parallelforall/cuda-pro-tip-kepler-shuffle/. Accessed 24 Feb 2018
7. De Leeuw J (1977) Applications of convex analysis to multidimensional scaling. In: Recent developments in statistics, North Holland Publishing Company, pp 133–145
8. Dzemyda G, Kurasova O, Žilinskas J (2013) Multidimensional data visualization: methods and applications, vol 75. Springer, Berlin
9. Dzwinel W, Blasiak J (1999) Method of particles in visual clustering of multi-dimensional and large data sets. Future Gener Comput Syst 15(3):365–379
10. Escobar R, Boppana RV (2016) Performance prediction of parallel applications based on small-scale executions. In: 2016 IEEE 23rd HiPC, pp 362–371
11. Fester T, Schreiber F, Strickert M (2009) CUDA-based multi-core implementation of MDS-based bioinformatics algorithms. In: Grosse I, Neumann S, Posch S, Schreiber F, Stadler PF (eds) GCB, LNI, vol 157. GI, Bonn, pp 67–79
12. Filatovas E, Podkopaev D, Kurasova O (2015) A visualization technique for accessing solution pool in interactive methods of multiobjective optimization. Int J Comput Commun Control 10:508–519
13. Garzón EM, Moreno JJ, Martínez JA (2017) An approach to optimise the energy efficiency of iterative computation on integrated GPU–CPU systems. J Supercomput 73(1):114–125
14. Goldberger J, Gordon S, Greenspan H (2003) An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. In: ICCV. IEEE Computer Society, pp 487–493
15. Hout MC, Goldinger SD, Brady KJ (2014) MM-MDS: a multidimensional scaling database with similarity ratings for 240 object categories from the massive memory picture database. PLoS ONE 9(11):1–11
16. Ingram S, Munzner T, Olano M (2009) Glimmer: multilevel MDS on the GPU. IEEE Trans Vis Comput Gr 15(2):249–261
17. Intel Math Kernel Library (Documentation) (2017) https://software.intel.com/en-us/mkl/documentation. Accessed 24 Feb 2018
18. Kurasova O, Petkus T, Filatovas E (2013) Visualization of pareto front points when solving multi-objective optimization problems. Inf Technol Control 42(4):353–361
19. Leng J et al (2013) GPUWatch: enabling energy optimizations in GPGPUs. SIGARCH Comput Archit News 41(3):487–498

20. Martínez JA, Almeida F, Garzón EM, Acosta A, Blanco V (2011) Adaptive load balancing of iterative computation on heterogeneous nondedicated systems. J Supercomput 58(3):385–393. https://doi.org/10.1007/s11227-011-0595-3
21. Martínez JA, Garzón EM, Plaza A, García I (2011) Automatic tuning of iterative computation on heterogeneous multiprocessors with ADITHE. J Supercomput 58(2):151–159
22. Medvedev V, Kurasova O, Bernatavičienė J, Treigys P, Marcinkevičius V, Dzemyda G (2017) A new web-based solution for modelling data mining processes. Simul Model Pract Theory 76:34–46
23. Morrison A, Ross G, Chalmers M (2003) Fast multidimensional scaling through sampling, springs and interpolation. Inf Vis 2(1):68–77
24. Orts F, Filatovas E, Ortega G, Kurasova O, Garzón EM (2017) HPC tool for multidimensional scaling. In: Vigo-Aguiar J (ed) Proceedings of the 17th international conference on computational and mathematical methods in science and engineering, vol 5, pp 1611–1614
25. Osipyan H, Morton A, Marchand-Maillet S (2014) Fast interactive information retrieval with sampling-based MDS on GPU architectures. In: Information retrieval facility conference. Springer, Cham, pp 96–107. ISBN: 978-3-319-12978-5
26. Papenhausen E, Wang B, Ha S, Zelenyuk A, Imre D, Mueller K (2013) GPU-accelerated incremental correlation clustering of large data with visual feedback. In: Proceedings of the 2013 IEEE international conference on big data, 6–9 Oct 2013, Santa Clara, CA, USA, pp 63–70
27. Park S, Shin SY, Hwang KB (2012) CFMDS: CUDA-based fast multidimensional scaling for genome-scale data. BMC Bioinform 13(17):S23
28. Pawliczek P, Dzwinel W, Yuen DA (2014) Visual exploration of data by using multidimensional scaling on multicore CPU, GPU, and MPI cluster. Concurr Comput 26(3):662–682
29. Qiu J, Bae SH (2012) Performance of windows multicore systems on threading and MPI. Concurr Comput Pract Exp 24(1):14–28
30. Shmoys DB, Tardos E (1993) An approximation algorithm for the generalized assignment problem. Math Program 62(3):461–474
31. Wong A, Rexachs D, Luque E (2015) Parallel application signature for performance analysis and prediction. IEEE Trans Parallel Distrib Syst 26(7):2009–2019
32. Yang T, Liu J, McMillan L, Wang W (2006) A fast approximation to multidimensional scaling. In: IEEE workshop on computation intensive methods for computer vision
33. Zhong Z, Rychkov V, Lastovetsky A (2014) Data partitioning on multicore and multi-GPU platforms using functional performance models. IEEE Trans Comput 64(9):2506–2518

2.4 Image Processing based on SMACOF and quantum computing

Two publications have been published in this area:

F. Orts, G. Ortega, E. Filatovas, O. Kurasova and E.M. Garzón. Hyperspectral Image Classification Using Isomap with SMACOF. *Informatica*, 30(2), 349-365, 2019. JCR (2019) = 3.312. Subject categories = Mathematics, Applied: 9/261 (**Q1**); Computer Science, Information Systems: 46/156 (Q2).

F. Orts, G. Ortega, A.C. Cucura, E. Filatovas and E.M. Garzón. Optimal fault-tolerant quantum comparators for image binarization. *Journal of Supercomputing*, 2021. JCR (2020) = 2.474. Subject categories = Computer Science, Theory & Methods: 33/110 (**Q2**); Computer Science, Hardware & Architecture: 26/53 (Q2); Engineering, Electrical & Electronic: 139/273 (Q3).

2.4.1 Hyperspectral Image Classification Using Isomap with SMACOF

Title	Hyperspectral Image Classification Using Isomap with SMACOF		
Authors	*F. Orts, G. Ortega, E. Filatovas, O. Kurasova and E.M. Garzón*		
Journal	Informatica		
Year	2019		
Volume	30(2)		
Pages	349-365		
DOI	https://doi.org/10.15388/Informatica.2019.209		
IF (JCR 2019)	3.312		
Categories	Mathematics, Applied:	9/261	**(Q1)**
	Computer Science, Information Systems:	46/156	(Q2)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

INFORMATICA, 2019, Vol. 30, No. 2, 349–365
© 2019 *Vilnius University*
DOI: http://dx.doi.org/10.15388/Informatica.2019.209

Hyperspectral Image Classification Using Isomap with SMACOF

Francisco José ORTS GÓMEZ[1]*, Gloria ORTEGA LÓPEZ[2],
Ernestas FILATOVAS[3], Olga KURASOVA[3],
Gracia Ester Martın GARZÓN[1]
[1]*Group of Supercomputation-Algorithms, Department of Informatics, University of Almería, ceiA3, 04120, Almería, Spain*
[2]*Computer Architecture Department, Campus Teatinos, Universidad de Málaga, 29010, Málaga, Spain*
[3]*Institute of Data Science and Digital Technologies, Vilnius University, Akademijos str. 4, LT-08663, Vilnius, Lithuania*
e-mail: francisco.orts@ual.es, gmartin@ual.es, gloriaortega@uma.es, ernest.filatov@gmail.com, olga.kurasova@mii.vu.lt

Received: October 2018; accepted: January 2019

Abstract. The isometric mapping (Isomap) algorithm is often used for analysing hyperspectral images. Isomap allows to reduce such hyperspectral images from a high-dimensional space into a lower-dimensional space, keeping the critical original information. To achieve such objective, Isomap uses the state-of-the-art MultiDimensional Scaling method (MDS) for dimensionality reduction. In this work, we propose to use Isomap with SMACOF, since SMACOF is the most accurate MDS method. A deep comparison, in terms of accuracy, between Isomap based on an eigen-decomposition process and Isomap based on SMACOF has been carried out using three benchmark hyperspectral images. Moreover, for the hyperspectral image classification, three classifiers (support vector machine, k-nearest neighbour, and Random Forest) have been used to compare both Isomap approaches. The experimental investigation has shown that better classification accuracy is obtained by Isomap with SMACOF.

Key words: dimensionality reduction, hyperspectral imaging, isometric mapping (Isomap), manifold learning, SMACOF algorithm.

1. Introduction

HyperSpectral Images (HSIs) contain an exhaustive variety of information about specific characteristics of the materials, with hundreds or even thousands bands (Borengasser *et al.*, 2007). The spectrum of each pixel can be seen as a vector, where each component represents the luminosity of the reflectance value for each spectral band. The set of bands which composes an HSI shows the representation of a scene, but each one individually contains information from a different wavelength range, which can cover both the visible

*Corresponding author.

and infrared spectrum. The width of each band can be between 5 and 10 nm, depending on the considered sensor. Each material throws a different reflectance profile for all the bands. Thus, for each point of the image, a specific curve that provides a lot of information for the corresponding point of the scene is obtained. Therefore, to efficiently exploit this information in applications, classification of HSIs is usually performed, where pixels are labelled to one of the classes based on their spectral characteristics.

There are many applications which take advantage of a large amount of information provided by hyperspectral sensors, such as remote sensing (Wang *et al.*, 2017), biotechnology (Asaari *et al.*, 2018), medical diagnose (Leavesley *et al.*, 2018), forensic science (Almeida *et al.*, 2017), environmental monitoring (Virlet *et al.*, 2017), etc. This available information leads us to develop new processing techniques. In addition, many applications which work with HSIs require a fast response. Examples of these applications may be obtained in the areas of modelling and environmental assessment, detection of military objectives or prevention and response to risks, such as forest fires, rescue operations, floods or biological threats (Chang *et al.*, 2001; Manolakis *et al.*, 2003).

However, the large amount of information contained in an HSI, which is its main advantage, is also a disadvantage in terms of computational performance. The work with large HSIs involves a high computational complexity and requires a lot of resources and time (Rizzo *et al.*, 2005). On the other hand, it is well-known that high-dimensional data spaces are mostly empty. This indicates that the data structure of an HSI exists basically in a subspace (Plaza *et al.*, 2005). Taking into account these ideas, it can be concluded that there is a need (and a possibility) to reduce the size of the HSIs. So, it is usual to apply techniques to reduce the dimensions of the original HSIs, obtaining reduced images which can be handled in a more efficient way without losing critical information (Harsanyi and Chang, 1994; Bruce *et al.*, 2002; Wang and Chang, 2006).

Multidimensional Scaling (MDS) consists of a set of techniques which are used to reduce the dimensions of a data set. Such techniques are used in many applications – multiobjective optimization (Filatovas *et al.*, 2015), data mining (Medvedev *et al.*, 2017), (Bernatavičienė *et al.*, 2007), marketing (Green, 1975), cryptography (Gupta and Ray, 2015), a wide variety of mathematical and statistical methods (Granato and Ares, 2014), psychology (Rosenberg, 2014), etc. They use a mapping function usually based on Euclidean distances which is able to find an optimal data representation. However, also other distance metrics could be considered (Fletcher *et al.*, 2014). MDS techniques represent data in a low-dimensional space in order to make these data more accessible (Borg and Groenen, 2005; Dzemyda *et al.*, 2013). For instance, a graphical visualization of the data in $2D$ or $3D$ space for an easier understanding of the information.

A well-known technique named Isometric mapping (Isomap) generalizes MDS to non-linear manifolds, replacing Euclidean distances by geodesic distances (Bengio *et al.*, 2004). Isomap has been used successfully in a multitude of applications, such as HSIs (Li *et al.*, 2017), face recognition (Yang, 2002a), biomedical datasets (Lim *et al.*, 2003), pattern classification (Yang, 2002b), learning multi-class manifold (Wu and Chan, 2004), supervised learning (Pulkkinen *et al.*, 2011), etc. Focusing on the HSIs, Isomap could be used in their reductions, achieving images with almost the same accuracy than the original but with fewer bands (Li *et al.*, 2017). The main goal here is to reduce the number of bands keeping the critical information they contain. Isomap is able to find hidden patterns in the bands and to reproduce the same pattern but with less bands.

Isomap often uses classical scaling such as eigen-decomposition as a part of its process. Classical scaling is a MDS method to reconstruct a configuration from the inter-point distance, which achieves a good accuracy and has a feasible computing cost (Sibson, 1979). However, any MDS method could be used.

The main contribution of this paper is the use of Isomap based on SMACOF (Scaling by MAjorizing a COmplicated), which is considered to be the most accurate MDS method (Borg and Groenen, 2005), and used when solving various MDS problems in social and behavioural sciences, marketing, biometrics, and ecology. Nevertheless, it is also one of the most computationally demanding methods (Ingram *et al.*, 2009). In previous work (Li *et al.*, 2017), where Isomap is studied in depth, authors consider classical scaling methods such as an eigen-decomposition process. However, our propose is to consider Isomap based on SMACOF due to its high accuracy. In this paper, the obtained results of both strategies, Isomap using eigen-decomposition and Isomap based on SMACOF, are compared in terms of classification accuracy. Such comparison is carried out by means of three popular HSIs and the same configurations in both cases.

The paper is organized as follows. In Section 2, the description of the Isomap method is provided. Section 3 describes the SMACOF algorithm. In Section 4, the results obtained after applying two versions of Isomap (with eigen-decomposition and with SMACOF) on several test images are discussed. Finally, we conclude this work in Section 5.

2. Isomap

Isomap is a manifold learning algorithm which can reduce the data redundancy preserving the original geometry of it. Isomap estimates the geodesic distance between all the items, given only input-space distances. For the points which are neighbours, input-space is an accurate approximation to the geodesic distance. For the distant ones, the geodesic distance can be computed as the addition of a sequence of distances between neighbouring points. The main idea is to find the shortest paths in a graph with edges connecting neighbouring data points (Tenenbaum *et al.*, 2000).

Isomap tries to build a matrix which contains all the minimum (geodesic) distances between the m items which are contained in a data set X (an HSI in our case), and then it reduces such matrix. In detail, the algorithm has three steps. They are shown in Algorithm 1 and described below:

1. To set a number l of neighbours. This number will be the same for all the items (points) X_i. Then, to determine the neighbours for every item X_i finding the l nearest points, taking into account that two points X_i and X_j cannot be neighbours if the distance between them is greater than a fixed value k. Euclidean distances between the m items are used. In this way, a graph G is constructed. Algorithm 2 describes the l-nearest neighbour (KNN) algorithm, which is commonly used to build neighbourhoods (Tay *et al.*, 2014).

```
Algorithm 1 Isomap($m, b, X, l, k, s, imax, \epsilon$)
Require:
    $m$: number of items;
    $b$: number of bands;
    $X$: $m \times b$ matrix which represents the HSI;
    $l$: maximum number of neighbours for each item;
    $k$: neighbourhood radius;
    $s$: dimension of low-dimensional space;
    $imax$: maximum number of iterations;
    $\epsilon$: threshold for the stress variance
Ensure:
    $Y$: set of finding points in the low-dimensional space stored in a $m \times s$ matrix
1: Construct the neighbourhood graph, $G = \text{KNN}(m, b, X, k, l)$ (Algorithm 2)
2: Compute shortest path between nodes, $\Delta = \text{Dijkstra}(m, G)$ (Algorithm 3)
3: Compute MDS method. For instance, SMACOF, $Y = \text{SMACOF}(m, s, \Delta, imax, \epsilon, Y)$ (Algorithm 4)
4: return $Y$
```

```
Algorithm 2 KNN($m, b, X, k, l, j$)
Require:
    $m$: number of items;
    $b$: number of bands;
    $X$: $m \times b$ matrix which represents the HSI;
    $k$: neighbourhood radius;
    $l$: maximum number of neighbours for each item;
    $j$: index of the selected item
Ensure:
    $G$: The neighbourhood graph
1: for $i = 0; i < m; i++$ do
2:     Compute Euclidean distances, $D = [d(X_i, X_j)]$
3: Compute set $G$ containing indices for the $l$ smallest distances (with $l < k$) of each element of $D$
4: return $G$
```

2. To calculate the shortest distance between all pair of points in G. When X_i and X_j are neighbours, their distance is Euclidean. However, when the points are not neighbours, the distance is computed as the shortest path between all possible ones in G which connects X_i and X_j, that is, $d(X_i, X_j) = min\{d_G(X_i, X_j), d_G(X_i, X_n) + d_G(X_n, X_j)\}$, where $n = 1, \dots, m$. As a result of this step, an $m \times m$ matrix which contains the short distances Δ, is obtained. In this work, Dijkstra's algorithm has been used to calculate the shortest paths among G according to Algorithm 3 (Dijkstra, 1959). Authors in Deng *et al.* (2012) explain Dijkstra's algorithm as these steps:
    - To initialize all nodes to ∞, except the initial, which is set to 0. Neighbours already have their distances. To mark all nodes as unvisited, as it is shown in Fig. 1(a).
    - To consider all the unvisited neighbours and to calculate their distances through each node. For every neighbour, to compare this distance with its previous distance and to assign the smallest one to the node. An example is shown in Fig. 1(b).

Algorithm 3 Dijkstra(m, G)

```
Require:
    $m$: number of items;
    $G$: neighbourhood graph
Ensure:
    $\Delta$: $m \times m$ matrix with the shortest distances
 1: for $i = 0; i < m; i{+}{+}$ do
 2:     $\Delta[i] = \infty$
 3:     $previous[i] = -1$
 4: $\Delta[0] = 0$
 5: $Q$ = the set of all nodes in $G$
 6: while $Q$ is not empty do
 7:     $u$ = vertex in $Q$ with the smallest $\Delta$
 8:     if $\Delta[u] = \infty$ then break;
 9:     Delete $u$ from $Q$
10:     for each neighbour $v$ of $u$ do
11:         $alt = \Delta[u] + distbetween(u, v)$
12:         if $alt < \Delta[v]$ then
13:             $\Delta[v] = alt$
14:             $previous[v] = u$
15:             Reorder $v$ in $Q$
16: return $\Delta$
```

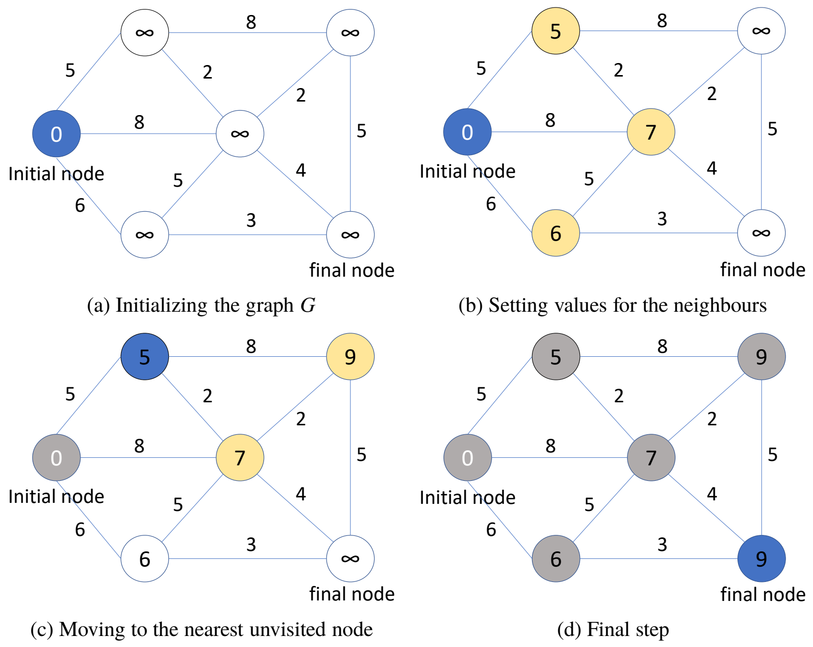
- When all the neighbours have been considered, to mark the current node as visited. A visited node will never be checked again. Move to the next unvisited node with the smallest distance and to repeat the previous steps, as it is shown in Fig. 1(c).
- If the final node has been marked as visited or if there is no path between the initial and the final node (all paths have a step marked as infinite), then the algorithm has finished. The final step is shown in Fig. 1(d).

3. To apply any MDS method to the shortest distances (Δ). Particularly, in this work, SMACOF and the eigen-decomposition methods are considered.

To evaluate the accuracy of Isomap based on SMACOF and eigen-decomposition methods for HSIs, a classification process with several classifiers – the Support Vector Machine (SVM) (Cortes and Vapnik, 1995), the KNN classifier (Altman, 1992) and the Random Forest algorithm (Breiman, 2001) – has been used.

3. SMACOF

SMACOF, as other MDS methods, is used for the analysis of similarity data on a set of items. As it has been mentioned before, SMACOF is the most accurate MDS technique (Ingram *et al.*, 2009). Its objective is to find a set of points $Y_1, Y_2, \dots, Y_m \equiv Y$ in a low-dimensional space $\mathbb{R}^s$, $s < b$ (where b is the original number of dimensions), taking into account that the distances between these points must be as similar as possible to the distance between the original points $X_1, X_2, \dots, X_m \equiv X$ (Orts *et al.*, 2018). The key is



(a) Initializing the graph G

(b) Setting values for the neighbours

(c) Moving to the nearest unvisited node

(d) Final step

Fig. 1. Steps of the Djikstra's algorithm.

the stress function (Eq. (1)). The less stress, the better results, since it measures the difference between the distances of the original points and the distances of the points in the low-dimensional space. In Eq. (1), δ represents the distance between points of X, and d does it between points of Y.

$$E_{MDS} = \sum_{i<j} \left(\delta_{ij} - d(Y_i, Y_j)\right)^2. \tag{1}$$

The majorizing concept, which implies to approximate a big or complex function through another smaller or simpler, is used by SMACOF to achieve the reduction of the stress (Groenen *et al.*, 1995). It consists of finding a new function iteratively. The new function will be located over the complex one, touching it at a point called *supporting* point (Fig. 2). Each iteration brings the minimum of the new function closer to the minimum of the original one, that is, the stress function (Borg and Groenen, 2005; Mairal *et al.*, 2014). In De Leeuw and Mair (2011), the majorization is defined in the following steps:

1. To choose an initial value $y = y_0$.
2. To find update x^t such that $g(x^t, y) \leqslant g(y, y)$.
3. If $f(y) - f(x^t) \geqslant \epsilon$, then $y = x^t$ and go to step 2.

In Algorithm 4, all the steps of SMACOF are shown. In such an algorithm, the initial value $y = y_0$ mentioned in step 1 is randomly generated. It has been tested in other works in which SMACOF obtains good results beginning from solutions randomly generated (Orts

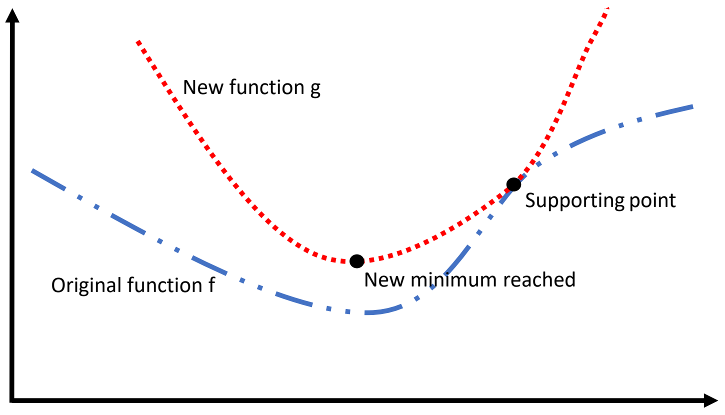


Fig. 2. Illustration of the majorization concept. The original function f is represented with a blue dashed line. The function obtained by majorization at every iteration, g, represented as a red dotted line, touches f at the supporting point. Taking into account that a new minimum of g is obtained at every iteration.

et al., 2018). The stress value of the current mapping is measured and then compared to the stress value of the previous mapping result. Each iteration minimizes the stress value due to the generation of closer solutions to the original. If the difference between the distances is smaller than a fixed threshold value, the algorithm stops (Ekanayake *et al.*, 2010), as it is mentioned in step 3. For the sake of simplicity, the details of the Guttman transform, used to update $x^{(t)}$, have not been explained here.

Algorithm 4 SMACOF($m, s, \Delta, imax, \epsilon$)

```
Require:
    $m$: number of items;
    $s$: dimension of low-dimensional space;
    $\Delta$: $m \times m$ matrix of dissimilarities of observed data on the high-dimensional space ($n$);
    $imax$: maximum number of iterations;
    $\epsilon$: threshold for the stress variance
Ensure:
    $Y$: set of finding points in the low-dimensional space stored in a $m \times s$ matrix
 1: Initial Solution randomly generated, $Y^0$
 2: **Compute Euclidean distances**, $D^0 = [d(Y_i^0, Y_j^0)]$
 3: $k = 0$, $error = 1$
 4: **if** ($k < imax$) and ($error > \epsilon$) **then**
 5:     **Compute Guttman transform matrix**, $B^k \equiv B^k(\Delta, D^{k-1})$
 6:     **Compute Guttman transform**, $Y^k = 1/m \cdot B^k \cdot Y^{k-1}$
 7:     **Update distances** $D^k = [d(Y_i^k, Y_j^k)]$
 8:     **Compute** $E_{MDS}^k$ (Eq. (1))
 9:     $error = |E_{MDS}^k - E_{MDS}^{k-1}|$
10:     $k = k + 1$
11: **return** $Y$
```

4. Evaluation Results

Such an investigation methodology has been considered in this work: first, to run Isomap based on SMACOF or eigen-decomposition methods and, after that, to apply a classification process with SVM, KNN or Random Forest classifiers.

Obtained results of Isomap using SMACOF are compared with the obtained results of a recent paper where Isomap considers an eigen-decomposition process (Li *et al.*, 2017) in the problem of hyperspectral images reduction. As in Li *et al.* (2017), three popular HSI images collected by the AVIRIS and ROSIS sensors have been considered to test Isomap (see Fig. 3). The considered data sets have the following characteristics:

- Pavia city centre (AVIRIS Salinas Valley, 2019), acquired by the ROSIS sensor. Pavia consists of 1096 × 715 pixels and 102 bands. For the sake of clarity, the data set is reduced to a 150 × 150 pixels subset. However, authors in Li *et al.* (2017) do not detail how they truncate the image in the study. In our work, random subsets of 150 × 150 are collected, keeping the ground truth variety.
- A finer spatial resolution of Salinas (AVIRIS sensor), named Salinas-A. Salinas-A consists of 86 × 83 pixels, which are the $[samples, lines] = [591 - 676, 158 - 240]$ of the original Salinas data set. It contains 204 bands.
- The Indian Pines data set (NW AVIRIS, 2012) collected by the AVIRIS sensor. It consists of 145 × 145 pixels and, originally, 224 bands. However, 24 bands which

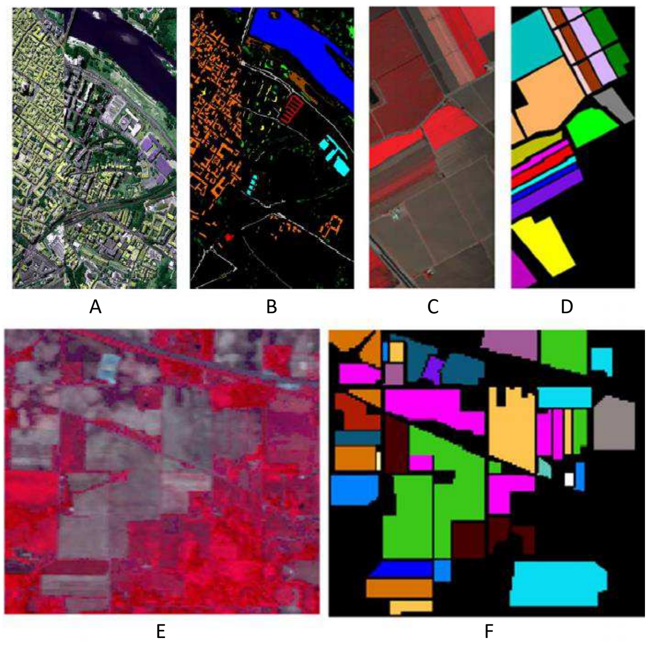


Fig. 3. HSIs tested. Pavia city centre (A) and its ground truth (B), Salinas-A (C) and its ground truth (D), and Indian Pines with its ground truth ((E) and (F) respectively).

contain the information about water absorption are removed in Li *et al.* (2017), so it has 200 bands in the tests.

Both Isomap versions (SMACOF and eigen-decomposition) have been implemented in Matlab and executed on a cluster composed by 64 cores of Bullx R424-E3 Intel Xeon E5 2650 with 8GB RAM. Specifically, KNN and Dijkstra procedures (Algorithms 2 and 3) have been coded using the Matlab functions *find_nn* and *dijkstra*, respectively. The precision of the classification process is dependent on the considered dimension of low-dimensional space (s) on Isomap. Therefore, several dimensions s have been taken into account to study their accuracy in the classification. Concretely, we varied the dimension of s from 10 to 50, as it was performed in Li *et al.* (2017). The parameter k, which describes the number of neighbours handled for each point has been set to 20.

We follow the idea described in Li *et al.* (2017) of considering several classifiers to evaluate the accuracy of both versions of Isomap for HSI classification, such as SVM and KNN classifiers. In addition, we have also considered the Random Forest algorithm. Similarly to Li *et al.* (2017), training and testing data were randomly selected from the ground truth. The 20% of the total pixels of each image were used to train, and the 80% to test. The comparative analysis has been based on the classification accuracy, which is obtained as the ratio: *correctly predicted data/total testing data*.

The SVM is coded using LIBSVM described in Chang and Lin (2011) with the following parameters: "$-t$ 2 $-c$ 100" ($-t$ 2 sets the type of kernel function as radial basis function, and $-c$ 100 set the cost parameter to 100). It is not necessary to set the gamma value, $-g$ (a parameter used as input by the radial basis function), as it is automatically set to "$-g$ $1/D$", where D is the dimension. The input data must be transformed following the data preprocessing described in Hsu *et al.* (2003). The results obtained using the SVM are depicted in Fig. 4.

KNN is a straightforward classification method, however, it is one of the most accurate ones (Keogh and Kasetty, 2002; Wei and Keogh, 2006). The results of the preliminary analysis of KNN are presented in Table 1 to consider the most suitable value of the number of neighbours (k'). This table shows the accuracy of the classification considering several values of k' $(1, 3, 5)$, for every reduced image on both dimensionality reduction methods (eigen-decomposition and SMACOF). Here, the best values are marked in italic style. As it can be observed in the table, the accuracy is reduced as the value of k' increases and 1NN obtains the best values of accuracy in all analysed cases. Therefore, KNN with $k' = 1$ (1NN) will be considered hereinafter. An additional advantage of 1NN is that it does not have tuning parameters and does not require a special transformation of the data or another preprocessing (Xing *et al.*, 2009). The Matlab function *fitcknn* has been used to perform KNN.

Apart from the classifiers used in Li *et al.* (2017), the Random Forest algorithm has also been considered in our evaluation (Fig. 6). The Matlab function *TreeBagger* has been used to perform Random Forest.

Obtained results with SVM, 1NN and Random Forest can be observed in Figs. 4, 5 and 6, respectively. The figures show the accuracy of the classification from the reduced images compared to the ground truth images, for both versions in each range from $s = 10$

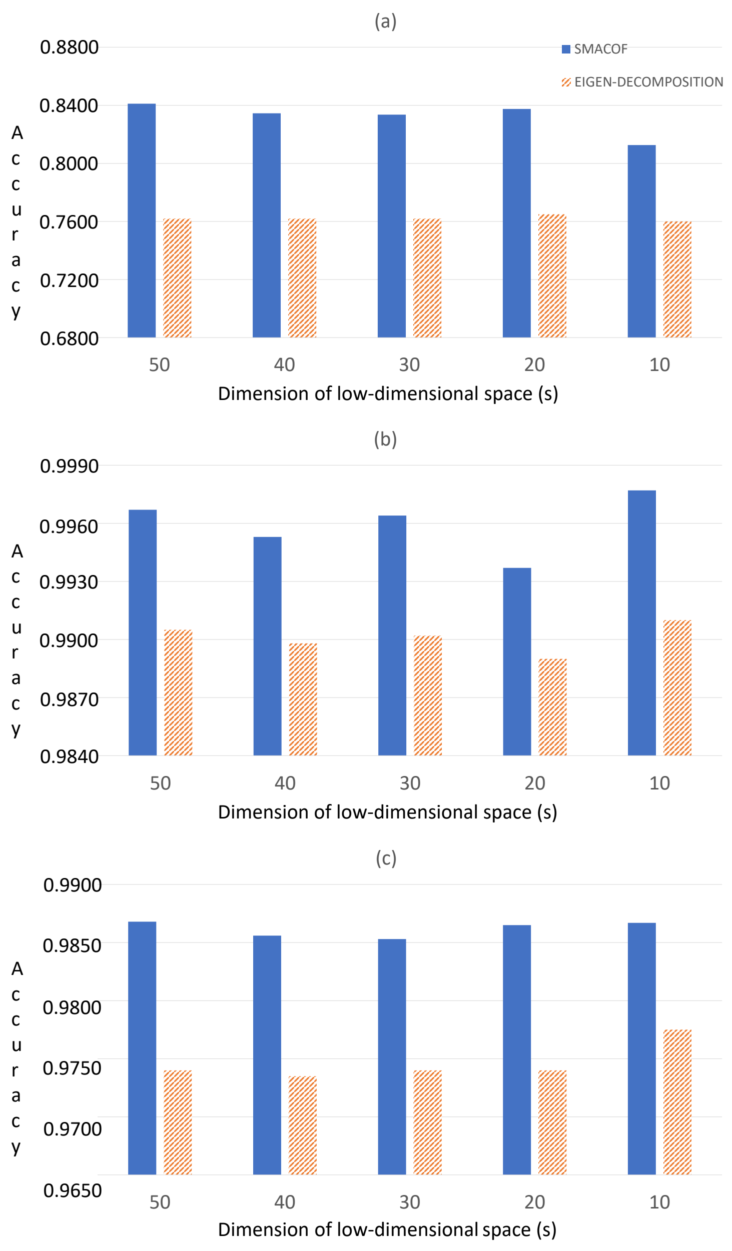


Fig. 4. Classification results (in terms of accuracy) of the three HSI data sets using SVM: (a) Indian Pines; (b) Salinas-A; (c) Pavia.

to $s = 50$. Such results have shown that the use of SMACOF improves the accuracy of Isomap for the three tested classifiers. In comparison with the version based on the eigen-decomposition process, the SMACOF approach is able to achieve better accuracies which involves a more optimized classification of HSI data sets.

Table 1
Classification results (in terms of accuracy) of the three HSI data sets using KNN for $k' = 1, 3$ and 5 and test images Indian Pines, Salinas-A and Pavia.

		SMACOF			EIGEN-DECOMPOSITION		
		k'					
IMAGE	s	1	3	5	1	3	5
Indian Pines	**50**	*0.8112*	0.7958	0.7943	*0.7250*	0.6956	0.6881
	40	*0.8046*	0.7987	0.7912	*0.7200*	0.6965	0.6884
	30	*0.8068*	0.7849	0.7814	*0.7150*	0.6933	0.6893
	20	*0.8179*	0.8069	0.7845	*0.7150*	0.6916	0.6879
	10	*0.8090*	0.7915	0.7877	*0.7050*	0.6896	0.6880
Salinas-A	**50**	*0.9946*	0.9931	0.9890	0.9899	0.9714	0.9658
	40	*0.9952*	0.9913	0.9925	*0.9896*	0.9733	0.9654
	30	*0.9950*	0.9935	0.9904	*0.9898*	0.9765	0.9645
	20	*0.9952*	0.9917	0.9914	*0.9892*	0.9743	0.9699
	10	*0.9963*	0.9890	0.9924	*0.9890*	0.9765	0.9687
Pavia	**50**	*0.9917*	0.9503	0.9488	*0.9729*	0.9365	0.9211
	40	*0.9929*	0.9407	0.9463	*0.9720*	0.9320	0.9232
	30	*0.9940*	0.9597	0.9525	*0.9729*	0.9365	0.9235
	20	*0.9937*	0.9598	0.9526	*0.9735*	0.9312	0.9245
	10	*0.9934*	0.9615	0.9576	*0.9715*	0.9348	0.9234

Once it is proven that the SMACOF approach is more accurate than the eigen-decomposition process, the global precision of Isomap with SMACOF has been tested in a more extended range of the values of s than (Li *et al.*, 2017) using 1NN (see Fig. 7). In this figure, it can be observed that the classification accuracy is quite high for all the analysed dimensionality reduction cases (from 50 to 2). However, it should be noted that the classification accuracy slightly decreases among the range from 9 to 2. Thus, we can conclude that SMACOF achieves a good accuracy even for the significant dimensionality reduction.

5. Conclusions

In this paper, our intention was to improve the accuracy of Isomap algorithm in the analysis of hyperspectral images. To achieve this, Isomap has been based on SMACOF, which is the most accurate MDS method, instead of classical scaling such as eigen-decomposition process.

The proposed version of Isomap based on SMACOF has been experimentally compared to a state-of-the-art version with an eigen-decomposition process. For that, well-known hyperspectral images taken from airbornes or satellites have been considered (Indian Pines, Salinas-A and Pavia Center). Moreover, a classification process using several classifiers (SVM, KNN and Random Forest) has been carried out to determine the accuracy of every test image with every method (SMACOF of eigen-decomposition). Obtained results have shown that the use of SMACOF improves the accuracy of Isomap in the reduction of the hyperspectral images for all studied cases.

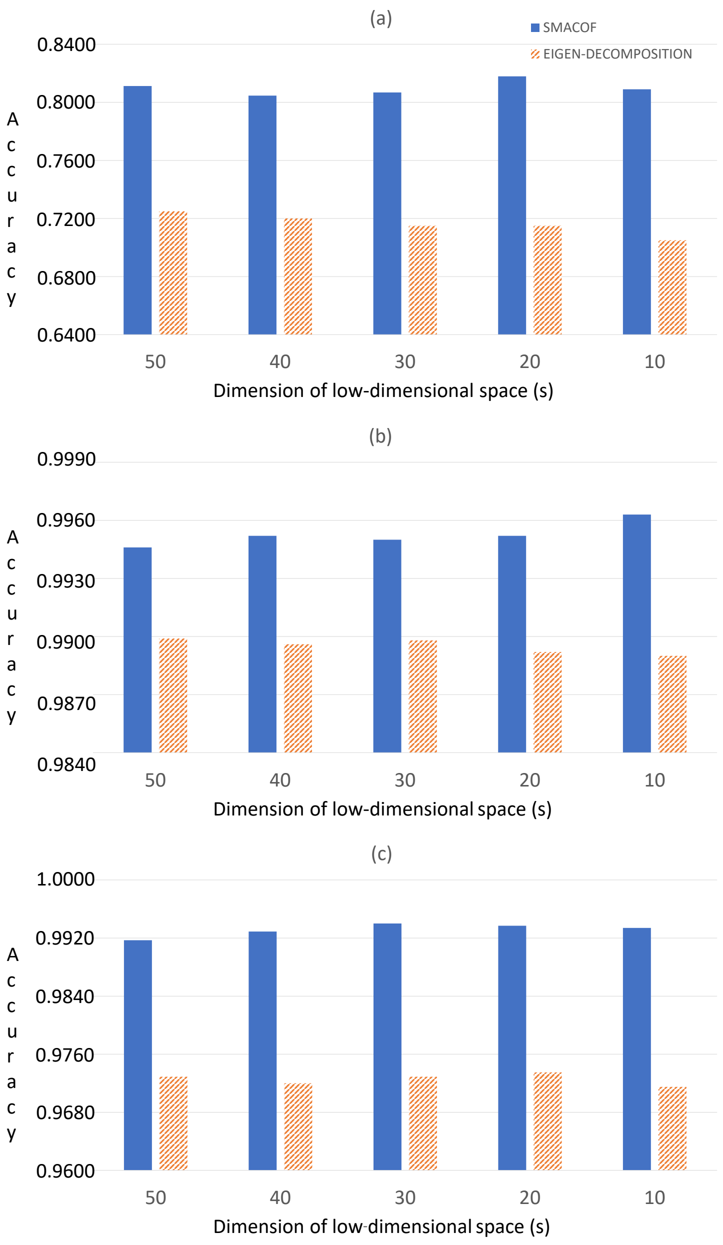


Fig. 5. Classification results (in terms of accuracy) of the three HSI data sets using 1NN: (a) Indian Pines; (b) Salinas-A; (c) Pavia.

In this work, only one criteria, the classification accuracy, is considered when reducing dimensions of the hyperspectral images. However, it should be noted that the drawbacks of Isomap and SMACOF are high consumptions of time and resources. Therefore, to de-

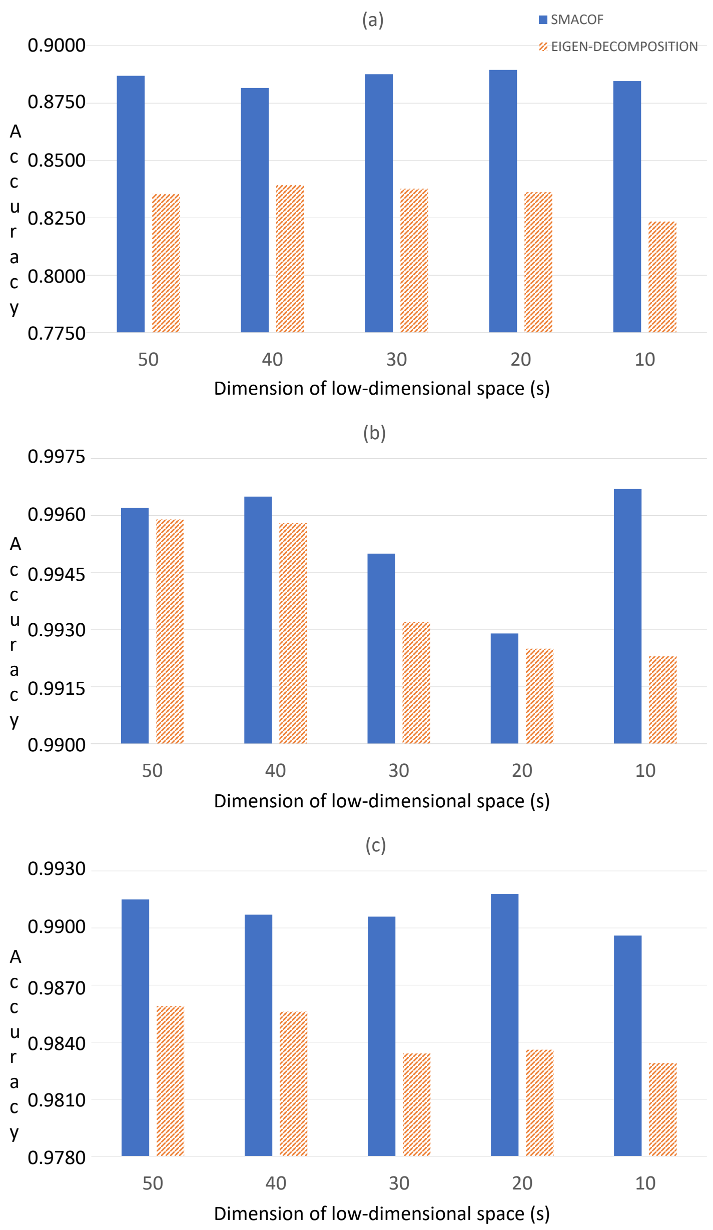


Fig. 6. Classification results (in terms of accuracy) of the three HSI data sets using Random Forest: (a) Indian Pines; (b) Salinas-A; (c) Pavia.

crease these aspects could be very valuable to make their application more approachable. Consequently, our current and future work is focused on the implementation of a GPU version of Isomap based on SMACOF.

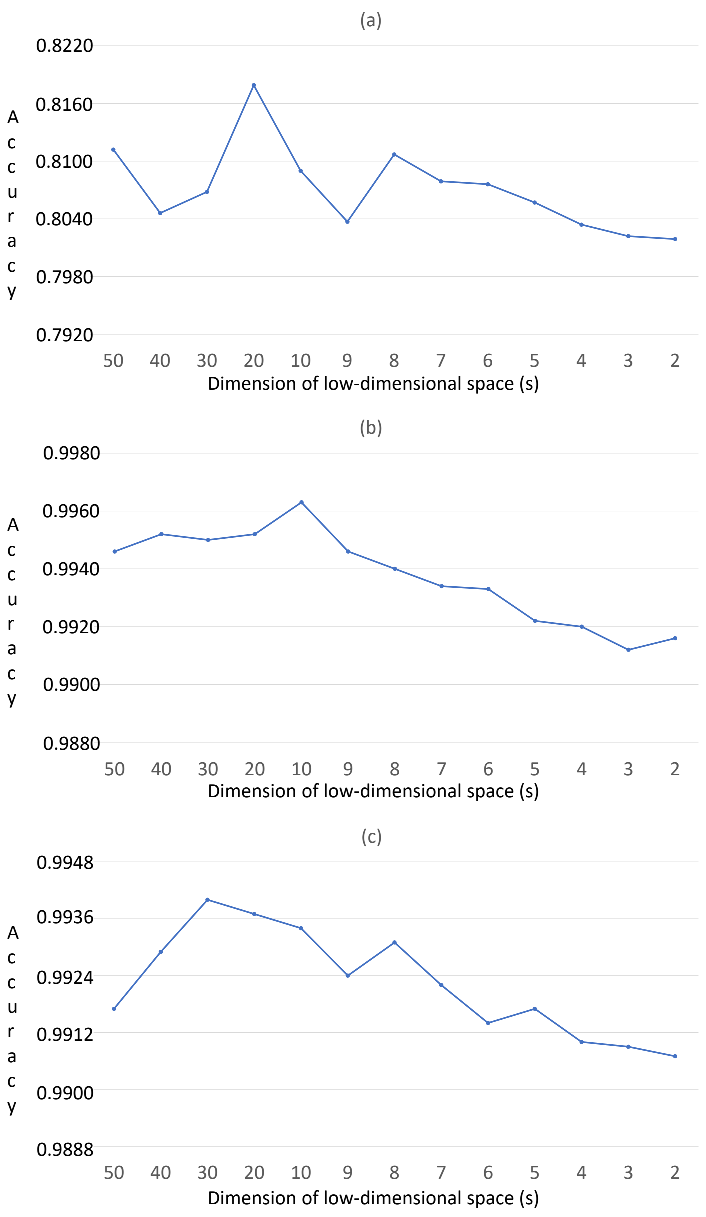


Fig. 7. Classification results (in terms of accuracy) of the three HSI data sets using 1NN for ranges from 50 to 2: (a) Indian Pines; (b) Salinas-A; (c) Pavia. Solid lines are to guide the eye.

Acknowledgements. This work has been supported by the Spanish Science and Technology Commission (CICYT) under contract TIN2015-66680; Junta de Andalucía under contract P12-TIC-301 in part financed by the European Regional Development Fund (ERDF). G. Ortega is a fellow of the Spanish 'Juan de la Cierva Incorporación' program.

References

Almeida, M., Logrado, L., Zacca, J., Correa, D., Poppi, R. (2017). Raman hyperspectral imaging in conjunction with independent component analysis as a forensic tool for explosive analysis: the case of an ATM explosion. *Talanta*, 174, 628–632.

Altman, N. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3), 175–185.

Asaari, M., Mishra, P., Mertens, S., Dhondt, S., Inzé, D., Wuyts, N., Scheunders, P. (2018). Close-range hyperspectral image analysis for the early detection of stress responses in individual plants in a high-throughput phenotyping platform. *ISPRS Journal of Photogrammetry and Remote Sensing*, 138, 121–138.

AVIRIS Salinas Valley (2019). Rosis pavia university hyperspectral datasets.

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Roux, N., Ouimet, M. (2004). Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In: *Advances in Neural Information Processing Systems*, pp. 177–184.

Bernatavičienė, J., Dzemyda, G., Marcinkevičius, V. (2007). Conditions for optimal efficiency of relative MDS. *Informatica*, 18(2), 187–202.

Borengasser M., Hungate W., Watkins R. (2007). *Hyperspectral Remote Sensing: Principles and Applications*. CRC Press.

Borg, I., Groenen, P. (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer Science & Business Media.

Breiman, L. (2001). *Random forests*. *Machine Learning*, 45(1), 5–32.

Bruce, L., Koger, C., Li, J. (2002). Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 40(10), 2331–2338.

Chang, C., Ren, H., Chiang, S. (2001). Real-time processing algorithms for target detection and classification in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(4), 760–768.

Chang, C., Lin, C. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 27.

Cortes, C., Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

De Leeuw, J., Mair, P. (2011). *Multidimensional Scaling Using Majorization: Smacof in R*.

Deng, Y., Chen, Y., Zhang, Y., Mahadevan, S. (2012). Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing*, 12(3), 1231–1237.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), 269–271.

Dzemyda, G., Kurasova, O., Žilinskas, J. (2013). *Multidimensional Data Visualization: Methods and Applications*. Springer.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S., Qiu, J., Fox, G. (2010). Twister: a runtime for iterative mapreduce. In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, pp. 810–818.

Filatovas, E., Podkopaev, D., Kurasova, O. (2015). A visualization technique for accessing solution pool in interactive methods of multiobjective optimization. *International Journal of Computers Communications & Control*, 10(4), 508–519.

Fletcher, R., Galiauskas, N., Zilinskas, J. (2014). Quadratic programming with complementarity constraints for multidimensional scaling with city-block distances. *Baltic Journal of Modern Computing*, 2(4), 248–259.

Granato, D., Ares, G. (2014). *Mathematical and Statistical Methods in Food Science and Technology*. John Wiley & Sons.

Green, P. (1975). Marketing applications of MDS: assessment and outlook. *The Journal of Marketing*, 24–31.

Groenen, P., Mathar, R., Heiser, W. (1995). The majorization approach to multidimensional scaling for Minkowski distances. *Journal of Classification*, 12(1), 3–19.

Gupta, K., Ray, I. (2015). Cryptographically significant MDS matrices based on circulant and circulant-like matrices for lightweight applications. *Cryptography and Communications*, 7(2), 257–287.

Harsanyi, J., Chang, C. (1994). Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach. *IEEE Transactions on Geoscience and Remote Sensing*, 32(4), 779–785.

Hsu, C., Chang, C., Lin, C. (2003). A practical guide to support vector classification, 1–16.

Ingram, S., Munzner, T., Olano, M. (2009). Glimmer: Multilevel mds on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 15(2), 249–261.

Keogh, E., Kasetty, S. (2002). On the need for time series data mining benchmarks: a survey and empirical demonstration. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 102–111.

Leavesley, S., Deal, J., Hill, S., Martin, W., Lall, M., Lopez, C., Boudreaux, C. (2018). Colorectal cancer detection by hyperspectral imaging using fluorescence excitation scanning. *Optical Biopsy XVI: Toward Real-Time Spectroscopic Imaging and Diagnosis*, 10489.

Li, W., Zhang, L., Zhang, L., Du, B. (2017). GPU parallel implementation of isometric mapping for hyperspectral classification. *IEEE Geoscience and Remote Sensing Letters*, 14(9), 1532–1536.

Lim, I., de Heras, P., Sarni, S., Thalmann, D. (2003). Planar arrangement of high-dimensional biomedical data sets by Isomap coordinates. In: *16th IEEE Symposium Computer-Based Medical Systems*, pp. 50–55.

Mairal, J., Bach, F., Ponce, P. (2014). Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2–3), 85–283.

Manolakis, D., Marden, D., Shaw, G. (2003). Hyperspectral image processing for automatic target detection applications. *Lincoln Laboratory Journal*, 14(1), 79–116.

Medvedev, V., Kurasova, O., Bernatavičienė, J., Treigys, P., Marcinkevičius, V., Dzemyda, G. (2017). A new web-based solution for modelling data mining processes. *Simulation Modelling Practice and Theory*, 76, 34–46.

NW Aviris (2012). *Indianas indian pines 1992 data set*.

Orts, F., Filatovas, E., Ortega, G., Kurasova, O., Garzón, E.M. (2018). Improving the energy efficiency of smacof for multidimensional scaling on modern architectures. *The Journal of Supercomputing*, 1–13.

Plaza, A., Martinez, P., Plaza, J., Perez, R. (2005). Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3), 466–479.

Pulkkinen, T., Roos, T., Myllymaki, P. (2011). Semi-supervised learning for wlan positioning. In: *International Conference on Artificial Neural Networks*. Springer, pp. 355–362.

Rizzo, F., Carpentieri, B., Motta, G., Storer, J. (2005). Low-complexity lossless compression of hyperspectral imagery via linear prediction. *IEEE Signal Processing Letters*, 12(2), 138–141.

Rosenberg, S. (2014). The method of sorting in multivariate research with applications selected from cognitive psychology and person perception. *Multivariate Applications in the Social Sciences*, 123–148.

Sibson, R. (1979). Studies in the robustness of multidimensional scaling: perturbational analysis of classical scaling. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 217–229.

Tay, B., Hyun, J., Oh, S. (2014). A machine learning approach for specification of spinal cord injuries using fractional anisotropy values obtained from diffusion tensor images. *Computational and Mathematical Methods in Medicine*.

Tenenbaum, J., De Silva, V., Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2319–2323.

Virlet, N., Sabermanesh, K., Sadeghi-Tehran, P., Hawkesford, M. (2017). Field Scanalyzer: An automated robotic field phenotyping platform for detailed crop monitoring. *Functional Plant Biology*, 44(1), 143–153.

Wang, J., Chang, C. (2006). Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 44(6), 1586–1600.

Wang, L., Zhang, J., Liu, P., Choo, K., Huang, F. (2017). Spectralspatial multi-feature-based deep learning for hyperspectral remote sensing image classification. *Soft Computing*, 21(1), 213–221.

Wei, L., Keogh, E. (2006). Semisupervised time series classification. In: *KDD 2006*, pp. 748–753.

Wu, Y., Chan, K. (2004). An extended Isomap algorithm for learning multi-class manifold. In: *Proceedings of 2004 International Conference IEEE Machine Learning and Cybernetics*, Vol. 6, pp. 3429–3433.

Xing, Z., Pei, J., Philip, S. (2009). Early prediction on time series: a nearest neighbor approach. In: *IJCAI*, pp. 1297–1302.

Yang, M. (2002a). Face recognition using extended Isomap. In: *IEEE Proceedings 2002 International Conference*.

Yang, M. (2002b). Extended Isomap for pattern classification. In: *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pp. 224–229.

F.J. Orts Gómez is a predoctoral researcher at the Informatics Department at University of Almería, Spain. He studied the master in computer engineering at the University of Almería. He is currently doing his PhD thanks to the Spanish FPI program. His publications and more information about him can be found in http://hpca.ual.es/~forts/. His research interests are multiDimensional scaling, quantum computation and high performance computing.

G. Ortega López (https://sites.google.com/site/gloriaortegalopez/) received the PhD degree from the University of Almería (Spain) in 2014. From 2009, she has been working as a member of the TIC-146 supercomputing-algorithms research group. Currently, she has a post-doctoral fellowship at the University of Málaga and her current research work is focused on high performance computing and optimization. Some of her research interest includes the study of strategies for load balancing the workload on heterogeneous systems, the parallelization of optimization problems and image processing.

E. Filatovas received the PhD in informatics engineering from the Vilnius University in 2012, Lithuania. He is currently a senior researcher at Vilnius University, and an associate professor at of Vilnius Gediminas Technical University. His main research interests include blockchain technologies, global optimization, multi-objective optimization, multi-objective evolutionary algorithms, multiple criteria decision making, high-performance computing, and image processing. He has published more than 20 scientific papers.

O. Kurasova received the doctoral degree in computer science (PhD) from Institute of Mathematics and Informatics jointly with Vytautas Magnus University in 2005. Recent employment is at the Institute of Data Science and Digital Technologies of the Vilnius University as a principal researcher and professor. Research interests include data mining methods, optimization theory and applications, artificial intelligence, neural networks, visualization of multidimensional data, multiple criteria decision support, parallel computing, image processing. She is the author of more than 70 scientific publications.

G.E.M. Garzón received her BSc degree in physics in 1985 from the University of Granada (Spain) and her PhD degree in computer engineering in 2000 from the University of Almería (Spain). She is a full-time professor at the Department of Informatics at the University of Almería. She is the head of the supercomputing-algorithms research group. Her research interest lies in the field of high performance computing for irregular problems related to image processing, matrix computation and optimization multicriteria.

2.4.2 Optimal fault-tolerant quantum comparators for image binarization

Title	Optimal fault-tolerant quantum comparators for image binarization		
Authors	*F. Orts, G. Ortega, A.C. Cucura, E. Filatovas and E.M. Garzón*		
Journal	Journal of Supercomputing		
Year	2021		
Volume	77 (8)		
Pages	8433-8444		
DOI	https://doi.org/10.1007/s11227-020-03576-5		
IF (JCR 2019)	2.474		
Categories	Computer Science, Theory & Methods:	33/110	**(Q2)**
	Computer Science, Hardware & Architecture:	26/53	(Q2)
	Engineering, Electrical & Electronic:	139/273	(Q3)

Contribution of the Ph.D. candidate

The Ph.D. candidate, F. Orts, is the first author and main contributor to this paper.

The Journal of Supercomputing
https://doi.org/10.1007/s11227-020-03576-5

Optimal fault-tolerant quantum comparators for image binarization

F. Orts[1] · G. Ortega[1] · A. C. Cucura[1] · E. Filatovas[2] · E. M. Garzón[1]

Accepted: 14 December 2020

Abstract
Quantum image processing focuses on the use of quantum computing in the field of digital image processing. In the last few years, this technique has emerged since the properties inherent to quantum mechanics would provide the computing power required to solve hard problems much faster than classical computers. Binarization is often recognized to be one of the most important steps in image processing systems. Image binarization consists of converting the digital image into a black and white image, so that the essential properties of the image are preserved. In this paper, we propose a quantum circuit for image binarization based on two novel comparators. These comparators are focused on optimizing the number of T gates needed to build them. The use of T gates is essential for quantum circuits to counteract the effects of internal and external noise. However, these gates are highly expensive, and its slowness also represents a common bottleneck in this type of circuit. The proposed quantum comparators have been compared with other state-of-the-arts comparators. The analysis of the implementations has shown our comparators are the best option when noise is a problem and its reduction is mandatory.

Keywords Quantum computing · Quantum image binarization · Quantum comparator

1 Introduction

Quantum computing has emerged as a new and promising science that has new challenges. One of them is that quantum computing is counterintuitive since it has some interesting but not intuitive features like entanglement and quantum parallelism [13]. Just a few years ago, the interest of researchers in quantum computing was focused on mathematical and physical fields because of the lack of real quantum computers and efficient quantum simulators. Recently, IBM, D-Wave, Google,

✉ G. Ortega
gloriaortega@ual.es

Extended author information available on the last page of the article

Published online: 21 January 2021

and other important organizations have built real and functional quantum computers [18]. Moreover, Microsoft, QuTech, Intel, Amazon and other vendors have opened new services based on several kinds of computers and architectures [5].

There are different paradigms in quantum computing. For instance, quantum annealers, like the D-Wave machine, are focused on solving problems that can be expressed as energy minimization [18]. On the other hand, topological quantum computers work with two-dimensional quasiparticles to process quantum information, which allows a better resilience against perturbations [17]. Nevertheless, topological quantum computers have not even been built nowadays, and only theoretical models have been developed. Furthermore, several ambitious quantum simulators have been developed recently, for example, QuEST, ProjectQ and myQLM [6, 8, 19].

Despite the fact that quantum technology is very innovative and powerful, there are many challenges to make quantum computing be practical. One of its main limitations is that quantum computers are difficult to program because their computational models are quite different from the classical one. The most well-known model is based on quantum circuits, where each specific procedure involves the design of particular quantum circuits. Because of the scarcity of quantum resources and the strong sensibility of the quantum computers to noise, the design of quantum circuits should be optimized in terms of number of resources and fault-tolerance. Therefore, an active research line is the optimal design of basic quantum operations involved in complex algorithms [14–16, 20].

Quantum image processing (QIMP) is an interdisciplinary subject between quantum computation and image processing. In recent years, along with the bright future of quantum computers, QIMP has become a hot research field. Combining quantum mechanics with image processing is an effective approach to improve the processing speed of images [21]. Its main function is to capture, manipulate and recover quantum images by means of the quantum computing [27]. According to the literature, QIMP techniques could improve classical processing algorithms in terms of performance, guaranteed security and minimal storage requirements [7, 27]. The benefits of such techniques have been demonstrated in a wide number of applications such as image classification, morphology, registration, synthesis, segmentation, filtering, and pseudocolor [28].

In this work, the focus is the quantum image binarization. The binarization is a crucial step in many image processing techniques. Binarization is a simple thresholding process over the image where the pixels with grey levels lower than a given threshold are classified into a class (i.e., the background), and all the remaining pixels into another (i.e., the foreground). It is well-known that the key of a binarization process is the comparison between each pixel and the threshold value. So, our intention is to design an efficient circuit to compare two quantum logic states and to identify whether they are equal or, otherwise, which of them is the largest [22].

There are already many classical methods proposed for image binarization [11]. However, quantum image processing provides an opportunity for faster image processing; therefore, recently received some attention in the quantum research community. Probably, the first proposed quantum comparator for image binarization is presented in [1]. A novel 8-bit half comparator was proposed in the context of binarization in [25], and in [26], it was optimized by rearranging the quantum gates. A quantum version of the Otsu's threshold selection method which contains image binarization procedure was designed in [10]. These publications showed that quantum computing offers a potential solution to efficiently deal with image binarization; however, currently, research content is very limited.

In this work, we propose two fault-tolerant comparators focused on optimizing the number of T gates. Quantum circuits are very sensitive to external and internal noise; therefore, noise reduction and fault tolerance are two of the most important objectives in quantum computing. The T gates are used to make possible the use of error-correcting codes to ensure fault-tolerance in quantum circuits. However, they are more expensive than other gates in terms of space and time cost due to, precisely, their increased tolerance to noise errors [12, 29]. In the design of quantum circuits, it is very relevant to specify the metrics used to evaluate the efficiency of such circuits. In order to evaluate our proposed and state-of-the-arts quantum circuits, we have considered the number of T gates a circuit has (T-count), the number of steps involving T gates, that is, the number of T gates which must be computed sequentially (T-depth) and the number of ancilla qubits.

The main contributions of the paper can be summarized as follows: (1) Development of a fault-tolerant quantum comparator; (2) Integration of the comparators in a circuit for image binarization that can be used as part of QIMP circuits that outperforms their classical counterparts [2, 3, 23]; and, (3) Evaluation of the proposed and other state-of-the-arts quantum comparators.

The manuscript is written as follows. Section 2 describes the quantum image binarization circuit design. In Sect. 3, we propose efficient quantum comparators. In Sect. 4, a comparative evaluation is carried out between our comparator and others of the state-of-the-art. Finally, we present the conclusions in Sect. 5.

2 Quantum image binarization circuit design

Our proposal is based on the binarization algorithm described by Xia et al [25]. This algorithm assumes that the image to be binarized is encoded in NEQR representation (Novel Enhanced Quantum Representation) [30]. In the NEQR representation, an image is represented according to the following equation:

$$|C_{YX}\rangle = \frac{1}{2^n} \sum Y = 0^{2^n-1} \sum X = 0^{2^n-1} |C_{YX}^{q-1} C_{YX}^{q-2} ... C_{YX}^{1} C_{YX}^{0}\rangle \otimes |YX\rangle \tag{1}$$

where $|C_{YX}^{q-1} C_{YX}^{q-2} ... C_{YX}^{1} C_{YX}^{0}\rangle$ codifies the value of the pixel (Y,X), n is related to the size of the image (it is a $2^n \times 2^n$ image), and q defines the color range as 2^q. YX encodes the spatial location of the pixel. A visual example of this representation for the 2^3 color range case is shown in Fig. 1.

For the sake of clarity, Algorithm 1 shows the pseudo-code for the 2^3 color range case. This algorithm needs two external values for each pixel of the image to be binarized: the codification $I = C_{YX}^{q-1} C_{YX}^{q-2} ... C_{YX}^{1} C_{YX}^{0}$ of the pixel, and a threshold value

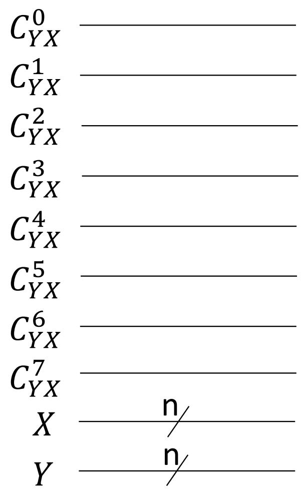


Fig. 1 A 2^3-color range image represented in NEQR. $C^7_{YX}...C^0_{YX}$ is the codification of the pixel, and XY the location

which is used to decide whether the pixel should be black or white. The algorithm consists of two steps:

- The first part compares C_{YX} with the threshold value b. There is no need to perform a complete comparison since it is only needed to compute if $C_{YX} < b$ or $C_{YX} \geq b$. Therefore, half comparator can be used. It should return $c = 1$ if $C_{YX} < b$, and 0 otherwise.
- The second part changes $C^{q-1}_{YX}C^{q-2}_{YX}...C^1_{YX}C^0_{YX}$ to 0 if $c = 1$, or to 1 if $c = 0$. That is, the algorithm sets the pixel as black if its original value is lesser than the threshold value, or sets it as white if its original value is greater or equal than the threshold value.

A quantum circuit to implement Algorithm 1 is shown in Fig. 2. On the one hand, the implementation of the second part of this circuit involves several swap gates. Such gates set the qubits of the pixel to 0 or 1 depending on the result of the comparison. The operation may seem simple, but it involves n inputs in state $|0\rangle$ and n in state $|1\rangle$. These states are swapped with the original pixel value under the conditions described in the previous paragraph. Also, and since we do not know beforehand into which group of inputs ($|0\rangle$ or $|1\rangle$) the original values of the pixels will be exchanged, we can therefore consider that we have $2n$ garbage outputs. On the other hand, the implementation of the half comparator is far from trivial [9, 24, 25]. This implementation is discussed in the next section.

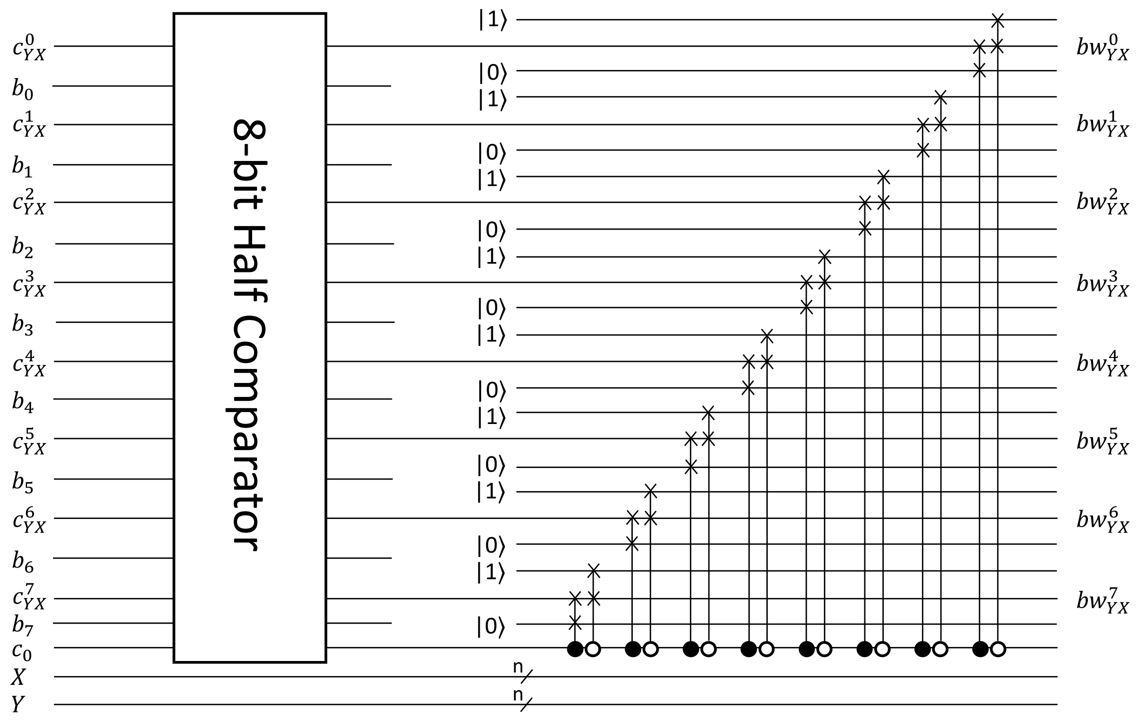


Fig. 2 Circuit implementation for Algorithm 1. This algorithm consists of two parts: a comparison between the pixel C_{YX} and the threshold b; and the assignment of the value 0 or 1 using swap gates to the pixel, according to the result of the previous comparison

Algorithm 1: Image binarization in quantum computing proposed in [25].

```
Result: A binary image $bw$.
$b = |b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\rangle$;
$I = |C^7_{YX} C^6_{YX} C^5_{YX} C^4_{YX} C^3_{YX} C^2_{YX} C^1_{YX} C^0_{YX}\rangle$;
$c_0 = |0\rangle$;
if $I < b$ then
    $c_0 = |1\rangle$;
end
Swap each quantum logic bit in $I$ with $c_o$;
$bw = I$;
return $bw$;
```

3 Proposed quantum comparators

In this section, we describe our proposed comparators. Two comparators have been developed as part of this work. The first one is focused on reducing the T-count, and the second comparator is focused on reducing the T-depth. They use the temporary logical-AND gate [4] in order to reduce the number of involved T gates. This gate performs an AND operation of two inputs (qubits), saving the result in an ancilla qubit. This is similar to the Toffoli gate, but the T-count of the temporary logical-AND is 4, and its T-depth is 2 (for the Toffoli gate, these

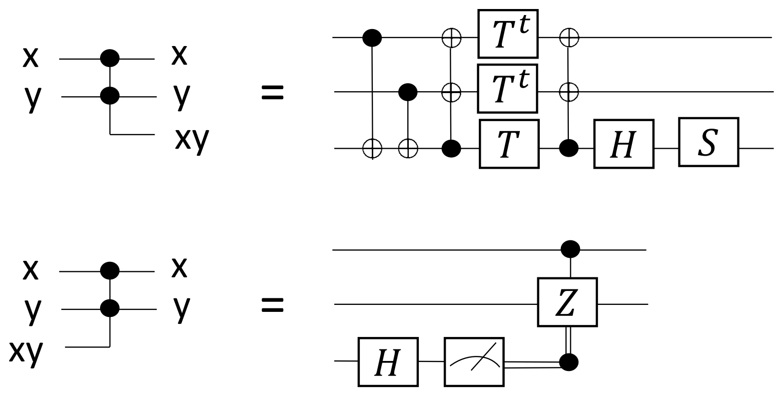


Fig. 3 Temporary logical-AND gate and its uncomputation

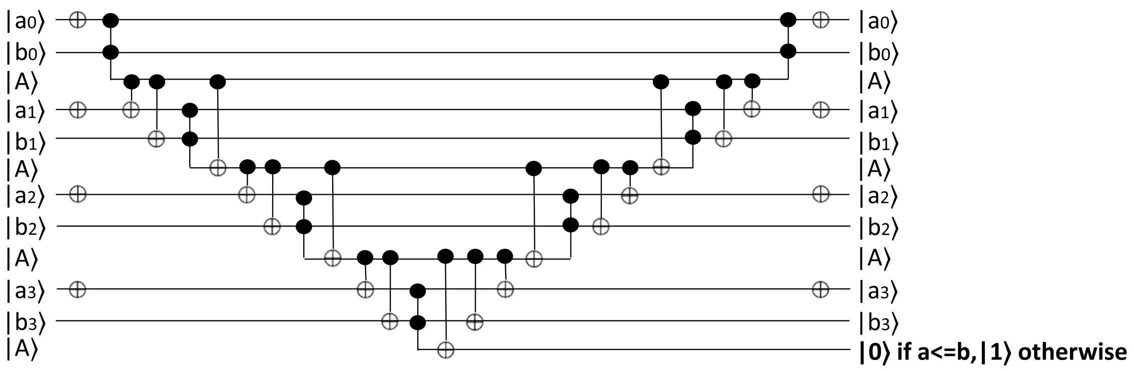


Fig. 4 Example of the first proposed comparator for the $n = 4$ case. This circuit is focused on reducing the T-count. a_i and b_i are the bit strings to be compared. A are ancilla qubits

values are 7 and 3, respectively). Moreover, the uncomputation of the temporary logical-AND does not involve T gates, whereas the uncomputation of the Toffoli gate involves another Toffoli gate. The temporary logical-AND gate (and its uncomputation gate) is shown in Fig. 3.

As it will be shown later, both of them have lower T-count and T-depth than existing quantum comparators.

The first proposed comparator is shown in Fig. 4. It is based on the methodology of the adder developed by Gidney in 2018 [4], which is the best adder in terms of T-count currently available [14]. The comparison between two-bit strings a and b is carried out performing the operation $a - b$. This operation can be performed using an adder, computing $\overline{\overline{a} + b}$. Actually, we are only interested in the sign of the operation, so that we can determine that a is lower than b if the sign of $a - b$ is negative, or that a is greater (or equal) than b if the result is positive. Therefore, several simplifications can be done to perform only the computation of the sign. The circuit can be reproduced for any size n of bits following these steps:

- For $i = 0$ to $i = n - 1$, to apply a Pauli-X gate at every bit a_i to perform $\overline{a}$. These operations are computed in parallel as it is shown in the circuit example of Fig. 4.
- Perform the operation $a_0 b_0$ using a temporary logical-AND gate instead of a Toffoli gate to save T-count and T-depth. Each temporary logical-AND will

require an extra qubit, which must be initialized to the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle)$. These ancilla qubits are marked as A in Fig. 4.

- For $i = 1$ to $i = n - 1$, apply two CNOT gates to compute $(a_{i-1}b_{i-1}) \oplus a_i$ and $(a_{i-1}b_{i-1}) \oplus b_i$. Then, to apply a temporary logical-AND to compute a_ib_i. Finally, to apply another CNOT gate to perform $(a_{i-1}b_{i-1}) \oplus (a_ib_i)$. Each step of the loop must be computed sequentially.
- The result is given by the last operation of the last iteration computed in the previous step. However, uncomputation is required to avoid garbage outputs. Applying two CNOT gates to perform $(a_{n-2}b_{n-2}) \oplus a_{n-1}$ and $(a_{n-2}b_{n-2}) \oplus b_{n-1}$.
- For $i = n - 2$ to $i = 1$, apply a CNOT gate to perform $(a_{i-1}b_{i-1}) \oplus (a_ib_i)$. Then, apply the uncomputation circuit for the logical and operation at a_ib_i. Finally, apply two CNOT gates at $(a_{i-1}b_{i-1}) \oplus a_i$ and $(a_{i-1}b_{i-1}) \oplus b_i$. Again, each step of the loop must be computed sequentially.
- Finally, for $i = 0$ to $i = n - 1$ apply a Pauli-X gate at every bit a_i to uncompute them. All the qubits except the one that contains the result have been uncomputed.

The second proposed comparator is shown in Fig. 5. It is based on the methodology of the adder developed by Thapliyal et al. in 2020 [20], which is the best adder in terms of T-depth currently available [14]. Again, the comparison is performed computing $\overline{a} + b$. This circuit involves the use of a huge amount of ancilla qubits to achieve a logarithmic T-depth since every and operation is performed using temporary logical AND gates. These gates could be replaced totally or partially to reduce the number of ancilla inputs. However, this will increase the T-depth and also the T-count of the circuit. The comparator can be reproduced for any size n of bits following these steps:

- For $i = 0$ to $i = n - 1$, to apply a Pauli-X gate at every bit a_i to perform $\overline{a}$. Then, apply a temporary logical-AND gate to calculate a_ib_i. According to the original adder, this value will be renamed as $g[i, i + 1]$.
- For $i = 1$ to $i = n - 1$, to apply a CNOT gate at $a_i \oplus b_i$. This value will be renamed as $p[i, i + 1]$.
- For $i = 2$ to $i = log(n) - 1$, and for $j = 1$ to $j = \frac{n}{2^i} - 1$, apply a temporary logical AND at locations $p[x,\ y]$, $p[y,\ z]$, being $x = 2^ij$, $y = 2^ij + 2^i$, and $z = 2^ij + 2^{i-1}$, respectively.
- For $i = 1$ to $i = log(n)$, and for $j = 0$ to $j = \frac{n}{2^i} - 1$, apply a temporary logical AND and an uncomputation gate at locations $g[x,\ y]$, $g[y,\ z]$, being $x = 2^ij$, $y = 2^ij + 2^i$, and $z = 2^ij + 2^{i-1}$, respectively.
- For $i = log(\frac{2n}{3})$ to $i = 1$, and for $j = 1$ to $j = \frac{n-2^{i-1}}{2^i}$, to apply a temporary logical-AND and its uncomputation at g[0,x], p[x,y] y g[x,y], being $x = 2^ij$, and $y = 2^ij + 2^{i-1}$, respectively.
- For $i = 1$ to $n - 1$, to apply a CNOT gate at p[i, i+1] and g[0, i].
- For $i = 1$ to $n - 1$, to apply a CNOT gate at p[0, 1] and the corresponding ancilla input.

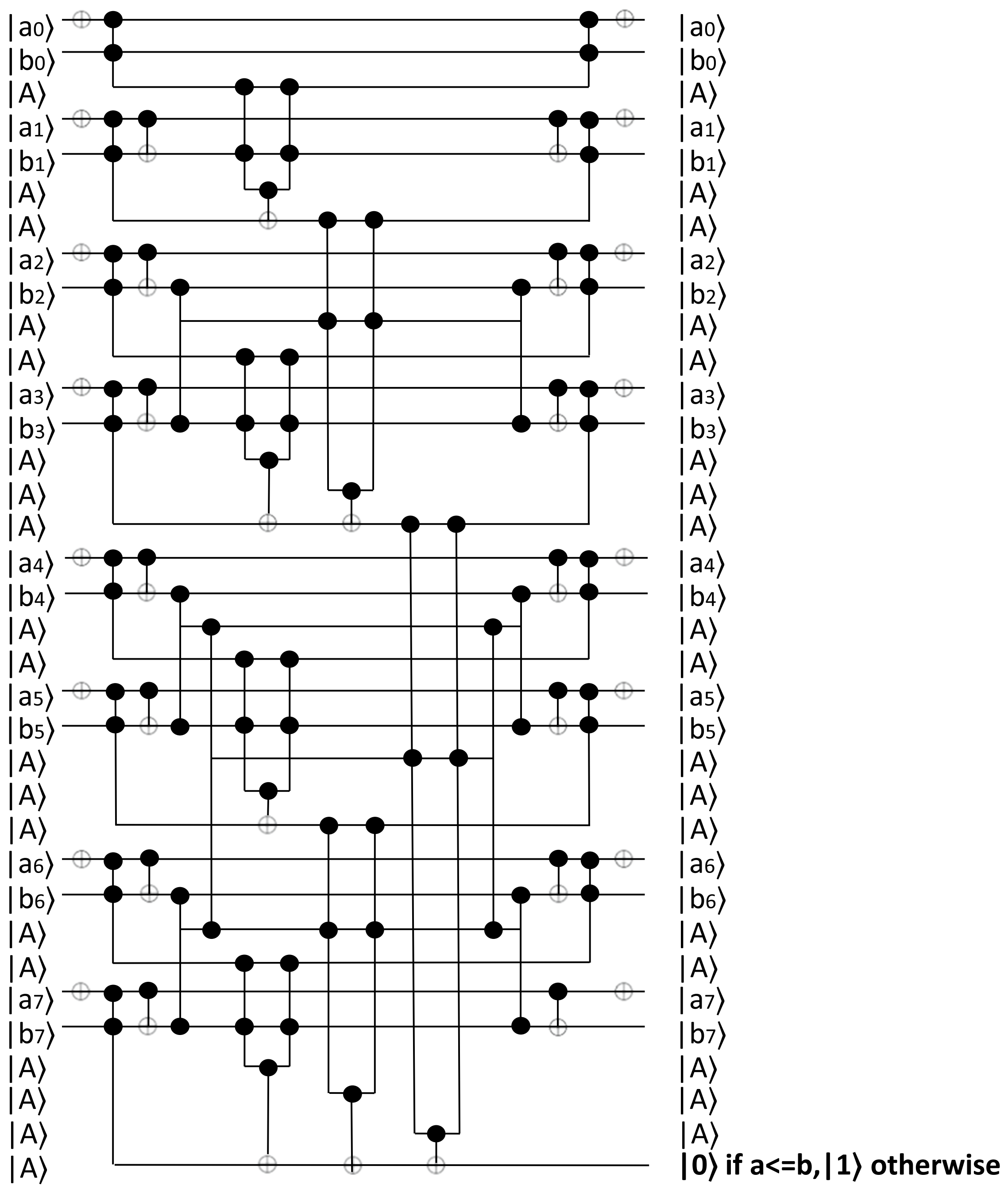


Fig. 5 Example of the second proposed comparator for the $n = 8$ case. This circuit is focused on reducing the T-depth. a_i and b_i are the bit strings to be compared. A are ancilla qubits

- Steps 3, 2, and 1 (in this order) must be computed again to uncompute garbage outputs.

4 Analysis and comparison

The proposed comparators consist of only four kinds of gates: Pauli-X gates, CNOT gates, temporary logical-AND gates, and the uncomputation gate for the temporary logical-AND gate. Among these gates, only the temporary logical-AND involves T gates. Therefore, the T-count and the T-depth of our circuits can be obtained from the total number of temporary logical-AND gates they have and the number of temporary logical-AND gates that the circuits must compute sequentially, respectively. The T-count and the T-depth of the temporary logical-AND gate are 4 and 2, respectively (Fig. 3).

The first circuit involves n consecutive temporary logical-AND gates. Then, it has a T-count of $4n$ and a T-depth of $2n$. Since the circuit only uses the ancilla qubits involved in the logical-AND operations, it can be concluded that the first comparator needs n ancilla qubits. On the other hand, the second comparator involves $3n - 2W(n) - log(n)$ temporary logical-AND gates, being $W(n)$ the number of ones in the binary expansion of n. Therefore, its T-count is $12n - 8W(n) - 4log(n)$. The T-depth is not trivial to compute since the depth of the circuit depends on the value of n. However, we have shown that the circuit grows logarithmically. Then, its T-depth can be set as $log(n)$.

Table 1 shows a comparison in terms of T-count, T-depth, and number of ancilla inputs between the most recent comparators in the state-of-the-art and our two proposed circuits. In terms of T-count and T-depth, it is shown that our circuits outperform the other comparators. Focusing on the T-count, the first proposed is the best option with a T-count of $4n$. The circuit with the best T-count in the literature is the proposal of Li et al. [9]. This circuit has a T-count of $14n - 7$, which is a value three times greater than our proposal. Our second proposal has a T-count of $12n - 8W(n) - 4log(n)$, which is still better than the circuit of Li et al [9].

Focusing now in the T-depth, the only logarithmic circuit is our second proposal. The other comparators are lineal. Again, the circuit of Li et al. is the best option in the literature, with a T-depth of $6n - 3$. Our first proposal, with a T-depth of $2n$, also outperforms the circuit of Li et al.

However, the circuit of Li et al. [9] has an important feature: it is the best in terms of necessary qubits. It is the only comparator with a single ancilla qubit. In these terms, our best proposal is the first one with n ancilla qubits. Therefore, the circuit of Li et al. [9] improves us in $n - 1$ qubits.

Table 1 Evaluation of comparators in terms of T-count, T-depth and Ancilla qubits as functions of n

Circuit cComparator	T-count	T-depth	Ancilla qubit
Xia et al. (2018) [24]	$14n$	$6n$	2
Xia et al. (2019) [25]	$14n - 7$	$6n - 3$	2
Li et al. (2020) [9]	$14n - 7$	$6n - 3$	1
Proposed comparator	$4n$	$2n$	n
Proposed comparator	$12n - 8W(n) - 4Log(n)$	$Log(n)$	$4n - 2W(n) - 2log(n)$

$W(n)$ is the number of ones in the binary expansion of n

5 Conclusion

In this paper, we continue the work started in [25] about binarization in quantum computing. In particular, we have improved a quantum circuit for binarization providing two novel comparators focused on the reduction of the internal and external noise. Although we work in a binarization framework, these comparators are valid for a general purpose.

Our two circuits are able to reduce the number of necessary T gates (which involves the reduction in the T-count and T-depth), thanks to the use of the temporary logical-AND gate proposed by [4], and also using the most efficient methodologies for noise reduction in quantum binary adders. Our first circuit is focused on the reduction in the T-count, and the second one is based on the reduction in the T-depth. However, the two circuits improve both in T-count and T-depth to the currently available circuits.

As a complement, we have carried out a comparison between our circuits and the most prominent comparators in the literature. The conclusions are that our circuits are the best option when noise is a problem and its reduction is mandatory. However, we also shown than the circuit proposed in Li et al. [9] is the best option when the focus is to minimize the number of qubits.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science throughout Project RTI2018-095993-B-I00 and by the European Regional Development Fund (ERDF).

References

1. Caraiman S, Manta V (2012) Image processing using quantum computing. In: 2012 16th International Conference on System Theory, Control and Computing (ICSTCC), pp. 1–6. IEEE
2. Chetia R, Boruah S, Roy S, Sahu P (2019) Quantum image edge detection based on four directional sobel operator. International Conference on Pattern Recognition and Machine Intelligence. Springer, Berlin, pp 532–540
3. Fan P, Zhou RG, Hu W, Jing N (2019) Quantum image edge extraction based on classical Sobel operator for NEQR. Quantum Inf Process 18(1):24
4. Gidney C (2018) Halving the cost of quantum addition. Quantum 2:74
5. Guerreschi GG, Hogaboam J, Baruffa F, Sawaya N (2020) Intel quantum simulator: a cloud-ready high-performance simulator of quantum circuits. CoRR **abs/2001.10554**
6. Häner T, Steiger DS, Svore K, Troyer M (2018) A software methodology for compiling quantum programs. Quantum Sci Technol 3(2):020501
7. Iliyasu AM (2013) Review towards realising secure and efficient image and video processing applications on quantum computers. Entropy 15:2874–2974. https://doi.org/10.3390/e15082874
8. Jones T, Brown A, Bush I, Benjamin SC (2019) Quest and high performance simulation of quantum computers. Sci Rep 9(1):1–11
9. Li HS, Fan P, Xia HY, Peng H, Long GL (2020) Efficient quantum arithmetic operation circuits for quantum image processing. Sci China Phys Mech Astronomy 63:1–13
10. Li P, Shi T, Zhao Y, Lu A (2020) Design of threshold segmentation method for quantum image. Int J Theor Phys 59(2):514–538
11. Michalak H, Okarma K (2019) Improvement of image binarization methods using image preprocessing with local entropy filtering for alphanumerical character recognition purposes. Entropy 21(6):562

12. Muñoz-Coreas E, Thapliyal H (2019) Quantum circuit design of a t-count optimized integer multiplier. IEEE Trans Comput 68(5):729–739
13. Nielsen MA, Chuang I (2002) Quantum computation and quantum information. Am J Phys 70:558
14. Orts F, Ortega G, Combarro EF, Garzón EM (2020) A review on reversible quantum adders. J Netw Comput Appl 170:102810. https://doi.org/10.1016/j.jnca.2020.102810
15. Orts F, Ortega G, Garzón EM (2019) An optimized quantum circuit for converting from sign-magnitude to two's complement. Quantum Inf Process 18(11):332. https://doi.org/10.1007/s11128-019-2447-7
16. Orts F, Ortega G, Garzón EM (2020) Efficient reversible quantum design of sign-magnitude to two's complement converters. Quantum Inf Comput 20(9–10):747–765
17. Pachos J, Lahtinen V (2017) A short introduction to topological quantum computation. SciPost Phys. https://doi.org/10.21468/SciPostPhys.3.3.021
18. Shin SW, Smith G, Smolin JA, Vazirani U (2014) How "quantum" is the D-Wave machine? arXiv preprint arXiv:1401.7087
19. Steiger DS, Häner T, Troyer M (2018) ProjectQ: an open source software framework for quantum computing. Quantum 2(49):10–22331
20. Thapliyal H, Muñoz-Coreas E, Khalus V (2020) T-count and qubit optimized quantum circuit designs of carry lookahead adder. arXiv preprint arXiv:2004.01826
21. Wang L, Ran Q, Ma J, Yu S, Tan L (2019) QRCI: a new quantum representation model of color digital images. Opt Commun 438:147–158
22. Wang D, Liu ZH, Zhu WN, Li SZ (2012) Design of quantum comparator based on extended general Toffoli gates with multiple targets. Comput. Sci. 39(9):302–306
23. Xia H, Xiao Y, Song S, Li H (2020) Quantum circuit design of approximate median filtering with noise tolerance threshold. Quantum Inf Process 19(6):183
24. Xia HY, Li H, Zhang H, Liang Y, Xin J (2018) An efficient design of reversible multi-bit quantum comparator via only a single ancillary bit. Int J Theor Phys 57(12):3727–3744
25. Xia HY, Li H, Zhang H, Liang Y, Xin J (2019) Novel multi-bit quantum comparators and their application in image binarization. Quantum Inf Process 18(7):229
26. Xia HY, Zhang H, Song SX, Li H, Zhou YJ, Chen X (2020) Design and simulation of quantum image binarization using quantum comparator. Mod Phys Lett A 35(09):2050049
27. Yan F, Iliyasu A, Le P (2017) Quantum image processing: a review of advances in its security technologies. Int J Quantum Inf 15:1730001. https://doi.org/10.1142/S0219749917300017
28. Yan F, Venegas-Andraca S (2020) Quantum image processing. Springer, Berlin
29. Zhang F, Chen J (2019) Optimizing t gates in Clifford+T circuit as $\pi/4$ rotations around Paulis
30. Zhang Y, Lu K, Gao Y, Wang M (2013) NEQR: a novel enhanced quantum representation of digital images. Quantum Inf Process 12(8):2833–2860

Authors and Affiliations

F. Orts[1] · G. Ortega[1] · A. C. Cucura[1] · E. Filatovas[2] · E. M. Garzón[1]

F. Orts
francisco.orts@ual.es

A. C. Cucura
acc166@inlumine.ual.es

E. Filatovas
ernest.filatov@gmail.com

E. M. Garzón
gmartin@ual.es

[1] Informatics Department, University of Almería, ceiA3, Carretera Sacramento s/n, Almería, Spain

[2] Institute of Data Science and Digital Technologies, Vilnius University, Akademijos str. 4, 08663 Vilnius, Lithuania

3. Other contributions

The research carried out on microrheology and dimensionality reduction has resulted in: i) 9 articles published in JCR journals (5 in Q1, 3 in Q2 and 1 in Q3), ii) another one under review, iii) another one in a non-indexed journal, iv) several contributions to international and national conferences, respectively, and v) one research stay.

Aside from its research, the Ph.D. candidate has also developed an extra research interest in Education. It is motivated by his active participation in teaching, namely, 119 hours during the realization of this thesis. The subjects taught encompassed Computer Architecture and Technology. As a result of the referred interest in Education, the Ph.D. candidate has also made contributions to national and international conferences.

Besides, the Ph.D. candidate has contributed reviewing several journal papers. All this information is included in the next sections.

3.1 Contributions to non-indexed journals

The candidate has published this non-indexed paper:

F. Orts, G. Ortega, E.M. Garzón. A faster half subtractor circuit using reversible quantum gates. Baltic Journal of Modern Computing, 2019.

3.2 Contributions to international conferences

The Ph.D. student has made the following contributions in international conferences:

E. Fernández-Combarro, I. Fernández-Rúa, F. Orts, G. Ortega, A.M.Puertas, and E.M. Garzón. On computing the neighbour list of the N-body Simulation by Quantum Algorithms, 21th International Conference on Mathematical Methods in Science and Engineering (CMMSE), 22-27 July 2021, Rota, Spain.

F. Orts, G. Ortega, E. Filatovas, and E. M. Garzón. Optimized 4-digits Quantum Carry Lookahead Adders, 27th International Conference on Parallel & Distributed Processing Techniques Applications (PDPTA), 26-29 July 2021, Las Vegas, USA.

F. Orts, A.C. Cucura, G. Ortega, E. Filatovas, E.M. Garzón. Image binarization using quantum computing, Proceedings of the 20th International Conference on Mathematical Methods in Science and Engineering, CMMSE 2020 - Rota, Spain.

F. Orts, A.C. Cucura, G. Ortega, E.M. Garzón, E.F. Combarro, I.F. Rúa. An optimized multi qubit quantum comparator, Quantum Computing Theory in Practice (QCTIP), 6-8 April 2020. Poster presentation. Cambridge, United Kingdom.

E. Filatovas, F. Orts, G. Ortega, O. Kurasova, E.M. Garzón. Acceleration of ISOMAP for Hyperspectral Image Classification on Multicore Processors and GPUs, 11th International Workshop on Data Analysys Methods For Software Systems (DAMSS), 28-30 November 2019. Poster presentation. Druskininkai, Lithuania.

F.J. Orts, G. Ortega, L.G. Casado, V. González-Ruiz, J.F. Sanjuan-Estrada. Quantum logic gates for students of computer engineering: a new learning method. 12th annual International Conference of Education, Research and Innovation (ICERI 2019), 11-13 November 2019, Seville, Spain.

J.J. Moreno, S. Puertas-Martín, F. Orts, N.C. Cruz, J.L. Redondo, E. Garzón, P.M. Ortigosa. On simulating an ARM processor for teaching computer structure. 12th annual International Conference of Education, Research and Innovation (ICERI 2019), 11-13 November 2019, Seville, Spain.

F. Orts, G. Ortega, N.C. Cruz, E.M. Garzón. Understanding Grover's search algorithm through a simple case of study 11th annual International Conference on Education and New Learning Technologies (EDULEARN19), 1-3 July 2019, Palma de Mallorca, Spain. (ISBN: 978-84-09-12031-4).

V. González, G. Ortega, E.M. Garzón, N.C. Cruz, J. Redondo, J. Salmerón, L. Casado, P.M. Ortigosa, C. Medina-López, J.J. Moreno, M. Ruiz-Ferrández, F. Orts, S. Puertas-Martín, T. Santamaría-López. Collaborative project-based learning: an experience 11th annual International Conference on Education and New Learning Technologies (EDULEARN19), 1-3 July 2019, Palma de Mallorca, Spain. (ISBN: 978-84-09-12031-4).

F. Orts, G. Ortega, E. Filatovas, O. Kurasova, E.M. Garzón. An efficient software for Hyperspectral Classification using Isometric Mapping, 10th International Workshop on Data Analysys Methods For Software Systems (DAMSS), 29 November-1 December 2018. Poster presentation. Druskininkai, Lithuania.

F. Orts, N.C. Cruz, S. Puertas-Martín, M. Ruiz-Ferrández, J.J. Moreno, C. Medina-López, P. Ortigosa, V. Ruíz, L. Casado, J.M. Salmerón, J.L. Redondo, E.M. Garzón, G. Ortega, R. Villegas. Learning Quantum Computation through simple examples. 11th annual International Conference of Education, Research and Innovation (ICERI 2018), 12-14 November 2018, Seville, Spain. (ISBN: 978-84-09-05948-5).

F. Orts, G. Ortega, E. Filatovas, O. Kurasova, E.M. Garzón. SMACOF algorithm to compress hiperspectral images, 16th EUROPT Workshop on Advances in Continuous Optimization, EurOpt 2018. 12-13 July, Almeria, Spain.

G. Ortega, F. Orts, A. Puertas, I. Fernandez, E.M. Garzón. A genetic solution for scheduling on unrelated heterogeneous parallel machines, 16th EUROPT Workshop on Advances in Continuous Optimization, EurOpt 2018. 12-13 July, Almeria, Spain.

F. Orts, G. Ortega, E.M. Garzón. A quantum circuit for solving divisions using Grover's search algorithm, Proceedings of the 18th International Conference on Mathematical Methods in Science and Engineering, CMMSE 2018 - Rota, Spain. (ISBN: 978-84-697-7861-6).

A.M. Puertas, F. Orts, G. Ortega, E.M. Garzón. Microrheology in hard colloids with large tracers, Joint Meeting of the DPG and EPS Condensed Matter Divisions, 11-16 Mars 2018. Oral presentation. Berlin, Germany.

G. Ortega, F. Orts, E. M. Garzón, E. Filatovas, O. Kurasova. CUDA and OpenMP Implementations for Solving SMACOF Problems, 9th International Workshop on Data Analysys Methods For Software Systems (DAMSS), 30 November-2 December 2017. Poster presentation. Druskininkai, Lithuania.

G. Ortega, E.M. Garzón, N.C. Cruz, J. Redondo, J. Salmerón, L. Casado, V. González, P. Ortigosa, C. Medina, J.J. Moreno, M. Ruíz, F. Orts, S. Puertas. Educational Strategies based on los cost platforms in the area of Computer Engineering. 10th annual International Conference on Education and New Learning Technologies (ICERI 2017), 16-18 November 2017, Seville, Spain. 7212-7218. 10.21125/iceri.2017.1928.

F. Orts, E. Filatovas, G. Ortega, O. Kurasova, E.M. Garzón. HPC Tool for Multidimensional Scaling, Proceedings of the 17th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2017 Vol. 5, pp. 1611-1614, 4-8 July 2017. Rota, Spain.

3.3 Contributions to national conferences

The student has made the following contributions in national conferences:

F. Orts, L.G. Casado, V. González-Ruíz, J.L. Redondo, G.M. Garzón, P.M. Ortigosa, J.J. Moreno, J.F. Sanjuan-Estrada, G. Ortega. Integración del currículo CISCO netacad como complemento docente V. Jornada de Innovación Docente y Experiencias Profesionales de la Universidad de Almería, 3 December 2020, University of Almería, Almería, Spain.

F. Orts, L.G. Casado, V. González-Ruíz, J.L. Redondo, G.M. Garzón, P.M. Ortigosa, J.J. Moreno, J.F. Sanjuan-Estrada, G. Ortega. Herramienta Kahoot para incrementar la participación de los alumnos en Arquitectura de Computadores. Jornada de Innovación Docente y Experiencias Profesionales de la Universidad de Almería, 3 December 2020, University of Almería, Almería, Spain.

F. Orts, G. Ortega, E.F. Combarro, E.M. Garzón. Revisión de sumadores cuánticos. IX Simposio de Investigación en Ciencias Experimentales 2020, 13 November 2020, University of Almería, Almería, Spain.

F. Orts. Un circuito cuántico optimizado para convertir de formato signo-magnitud a complemento a dos. III Jornadas de Doctorado en Informática (JDI2020@UAL), 21 February 2020, University of Almería, Almería, Spain.

F. Orts. Revisión y optimización de circuitos cuánticos. XI Seminario de Invierno CAPAP-H, 6-7 February 2020, Universitat Autònoma de Barcelona, Barcelona, Spain.

F. Orts, G. Ortega, E.M. Garzón. Building blocks for quantum circuits. VIII Simposio de Investigación en Ciencias Experimentales, 14-15 November 2019, Universidad de Almería, Almería, Spain. p. 56. University of Almería, ISBN 978-84-17261-95-5 .

F. Orts , S. Puertas Martín, J.J. Moreno, N.C. Cruz, J.L. Redondo, E.M. Garzón, P.M. Ortigosa, G. Ortega, M.R. Ferrández, C. Medina López, V. González Ruíz, L.G. Casado, J.M.G. Salmerón, T.M. Santamaría, J.F. Sanjuan-Estrada. Simulación de un procesador ARM para la enseñanza de Estructura de Computadores. Póster en Jornadas sobre Innovación Docente 2019-2020. 19 September 2019. University of Almería, Spain.

F. Orts, G. Ortega, E.M. Garzón. Diseño de un semirrestador cuántico eficiente. XXX Jornadas de Paralelismo (JP2019), 18-20 September 2019, Cáceres, Spain.

G. Ortega, F. Orts, A.M. Puertas, I. Fernández, E.M. Garzón. Scheduling paralelo sobre clústeres heterogéneos: Microreología Activa como caso de estudio. XXX Jornadas de Paralelismo (JP2019), 18-20 September 2019, Cáceres, Spain.

S. Puertas-Martín, J.J. Moreno, F. Orts, N.C. Cruz, J. Redondo, E.M. Garzón, P.M. Ortigosa. Solución de un procesador ARM para la enseñanza de Estructura de Computadores. XXX Jornadas de Paralelismo (JP2019), 18-20 September 2019, Cáceres, Spain.

F. Orts. Optimizando la eficiencia energética de SMACOF. II Jornadas de Doctorado en Informática (JDI2019@UAL), 14 February 2019, University of Almería, Almería, Spain. pp. 77-86.

F. Orts, G. Ortega, E.M. Garzón. Generating quantum circuits for solving algebraic equations using Grover's algorithm. VII Simposio de Investigación en Ciencias Experimentales. 15-16 November 2018. University of Almería, Spain.

F. Orts, G. Ortega, N.C. Cruz, S. Puertas-Martín, M.R. Ferrández, J.J. Moreno, C. Medina-López, P.M. Ortigosa, V. González-Ruíz, L.G.Casado, J.M.G. Salmerón, J.L. Redondo, R.

Villegas, T.M. Santamaría, J.F. Sanjuan-Estrada, E.M.Garzón. Recursos para el aprendizaje de computación cuántica en el Grado de Ingeniería Informática. Póster en Jornadas sobre Innovación Docente 2018-2019. 19 September 2018. University of Almería, Spain.

F. Orts, E. Filatovas, G. Ortega, O. Kurasova, E.M. Garzón. Mejorando la eficiencia energetica de SMACOF en arquitecturas modernas. XXIX Jornadas de Paralelismo (JP2018), 12-14 September 2018, Teruel, Spain.

F. Orts. Acelerando la computación científica basada en estructuras de datos irregulares. I Jornadas de Doctorado en Informática (JDI2018@UAL), 15 February 2018, University of Almería, Almería, Spain. pp. 79-88.

F. Orts, G. Ortega, A.M. Puertas, E.M. Garzón. Planificando un problema de microreología en un clúster Multi-GPU. XXVIII Jornadas de Paralelismo (JP2017), 19-22 September 2017, Málaga, Spain.

G. Ortega, J.J. Moreno, E.M. Garzón, N.C. Cruz, J. Redondo, J. García, L. González, V. González, P. Ortigosa, C. Medina, M. Ruíz, F. Orts, S. Puertas. Utilizando recursos de bajo coste en asignaturas de Ingeniería Informática. Póster en Jornadas sobre Innovación Docente 2017-2018. 21 September 2017. Universidad de Almería, Spain.

G. Ortega, F. Orts, A.M. Puertas, E.M. Garzón. Acceleration of a colloidal microrheology model based on CUDA. V Simposio de Investigación en Ciencias Experimentales, 15 November 2016, Universidad de Almería, Almería, Spain. pp. 40-40. University of Almería, ISBN 978-84-16642-49-6.

3.4 Contributions as a reviewer

The candidate has been reviewed papers for 5 different journals:

- Journal of Ambient Intelligence and Humanized Computing
- ACM Transactions on Knowledge Discovery from Data
- Computational and Mathematical Methods
- Journal of Computational and Applied Mathematics
- The Journal of Supercomputing

4. Conclusions and future work

In this thesis, new approaches for solving computational problems in three different lines using HPC and quantum computing have been proposed and studied. The first line of research focuses on making computationally feasible the reproduction of a series of simulations in the field of microrheology. The second research line deals with MultiDimensional Scaling methods and their high computational cost. The third one is about Image Processing. From the computational point of view, the three lines have been approached with a similar methodology, applying from classical HPC techniques to new quantum computing techniques. The application of HPC techniques has improved the performance and energy efficiency of the problems studied. In addition, the solutions to these problems have been extended. In the context of quantum computing, we have focused on the design of optimized quantum circuits to solve the corresponding problems. The main problems have been formulated in terms of mathematical optimization. Therefore, a state-of-the-art study on the possible tools and methods that could be applied for their solution has been carried out. Then, the most appropriate tool was chosen for each case, adapting and optimizing it where necessary, and even combining tools or innovating in cases where no available tool provided the necessary solution. The use of parallel computing and quantum computing has been considered during the whole thesis as a way to improve the developed methods. This chapter draws the main conclusions in each line.

4.1 Microrheology

At the physical level, important advances have been made in the extension of the size of the model-based simulations. The model has been strongly expanded thanks to the new development of the microrheology model [54] since such a development allows the exploitation of the parallelism levels of the model on modern clusters. An enormous amount of simulations have been carried out to generate sufficient statistics for such a definition, which has been possible thanks to the acceleration carried out at each of the three possible levels of parallelism: the calculation of a trajectory, the computation of all trajectories of a given size, and the calculation of the various sizes have been optimized to finally extrapolate the friction coefficient.

Going into more detail on the computational achievements, a GPU version of the trajectory calculation has been implemented. This version made in CUDA achieves such acceleration thanks to the reordering of the different data structures for its optimal treatment in the GPU, as well as with different kernels that through tools like cuBLAS and the shuffle instructions manage to perform the calculations minimizing the necessary shared memory and the CPU-GPU information transfer.

On the other hand, a genetic algorithm customization has been proposed to harness the heterogeneous resources of the modern cluster when all sets of simulations are computed to extrapolate the friction coefficient. In particular, this genetic algorithm, by means of the appropriate definition of the usual operators in this type of algorithms, manages to find the appropriate distribution to assign each of the trajectories (for several system sizes) to be calculated to the corresponding machine so that the total time required for the calculation of all the trajectories is the shortest possible. To this end, the algorithm focuses on load balancing reduction, using two versions of the program and all the processing units of our multi-GPU cluster, although it is fully valid for any number of programs and processing units.

Finally, up to three functional quantum algorithms have been defined to speed up one of the most expensive parts of the simulations: the calculation of neighbors. These three algorithms form a meta-algorithm that, depending on the starting conditions, allows choosing which of these three algorithms should act to perform the neighbor search to reduce the computational needs, identified in this case as calls to the oracle. Concerning the latter, a fully functional circuit for the calculation of distances between particles and a comparison of these distances with a threshold value (which is precisely the one that defines which particles are or are not neighbors) has been developed.

4.2 MultiDimensional Scaling

In reference to this line, we have made two contributions. First, there are our two versions of SMACOF, one multicore and the other for GPU. SMACOF is one of the most accurate of the MDS methods, but also the most computationally expensive. However, our versions considerably speed up its computation, making SMACOF a more affordable method. The multicore version, made in C and OpenMP, is supported by the MKL library. The GPU version in C and CUDA, and uses the cuBLAS library. Both versions optimize the processing of the various data structures and operations to achieve a further reduction in computation time.

On the other hand, a heuristic oriented to the optimization of energy consumption has been defined. Since there are two versions of SMACOF, what this heuristic does is to estimate the energy efficiency of each version in each of the processing units available in one or more clusters. By means of a quick estimation, it provides information that not only allows to choose the appropriate cluster for the user's needs, but also the appropriate configuration of such cluster to minimize the energy consumption.

4.3 Image Processing

A successful attempt has been made to apply ISOMAP for the classification of so-called hyperspectral images. ISOMAP is a method consisting of three distinct parts, one of them being an MDS method. Although SMACOF is the most accurate MDS method, it is not commonly used due to its high computational cost, but thanks to our advances in the previous line we have used SMACOF. The use of ISOMAP with SMACOF has allowed us to improve the classification accuracy in hyperspectral images, as demonstrated by the different tests we have performed on some of the most commonly used images for testing and demonstration.

Another part of ISOMAP is very similar to the neighbor search computation discussed in the microrheology line. That is why we want to address its resolution by quantum computation. Although we have not yet been able to do so given the lack of tools for digital image processing in quantum computing, we have begun to design various tools that will allow us to achieve this goal. We have successfully designed an image binarizing circuit that improves on those currently available in the literature in terms of fault tolerance, which is a first step towards the aforementioned goal.

4.4 Quantum Circuits

As part of the thesis, numerous contributions have been made to the field of quantum computing, in the specific design of quantum circuits. One of these contributions has consisted of a wide review of quantum adder circuits, considered the most important arithmetic circuits in quantum computing today because of their relationship with Shor's algorithm. This review previously defined a solid measurement platform to provide complete and uniform information on each of the circuits reviewed. Once the metrics have been chosen, more than 40 bibliographic references have been reviewed, their circuits have been analyzed, and they have been classified by typology and metric.

Significant progress has also been made in terms of two's complement converters. We have shown that this numerical representation also has its advantages in quantum computing. Several converters circuits have been proposed, each of them being the best in its category: the first of them is the fastest, and the others need the lowest quantum cost and/or auxiliary inputs. Both circuits, like all the circuits we make, are free of garbage outputs to allow all qubits to interleave with other circuits.

We have also made contributions in the field of comparators through the realization of several of these circuits. Our comparators have proven to be the best of their kind in terms of fault tolerance thanks to a proper use of T-gates that allow performing operations properly while minimizing cost and computation time. These circuits have allowed advances in image binarization, as indicated in the image processing part.

A functional oracle has also been developed to indicate whether two particles are closer or not to a threshold distance. This circuit has two distinct parts: the calculation of distances between particles, and the comparison of this distance with the threshold value. This second part can be performed with any comparator, including ours. However, in the distance calculation there are several operations involved (adders, subtractors, squares), so a review of the state of the art circuits available for each type of operation (fortunately we already had the review of adders) had to be carried out to find the optimal circuit for these operations.

4.5 Future work

As future work, it is possible to define new goals in the three main research lines covered. Focusing on Microrheology, we will continue extending the model. We are currently studying the obtained results of our simulations in different scenarios in order to test the applicability of the model in them. We will also analyze new improvements and extensions to the related code using the latest GPU technologies. Special attention will be paid to quantum computing. Our goal is to apply our new circuits (and design ad-hoc ones) to accelerate the collection of statistics on the models results. Finally, we will consider advancing the theory on microrheology to empirically test certain assumptions that have only been demonstrated at the mathematical level.

In relation to MultiDimensional Scaling methods, we would like to extend some of the results achieved with quantum computation in the other two lines to this field, both for the individual advancement of this line and for the one related to hyperspectral imaging. At another level, we would like to realize a CUDA version of the well-known Glimmer algorithm (another MDS method, currently available only in OpenCL), to make a comparison between SMACOF and it and to be able to include it in our energy efficiency heuristics. We have also superficially tested the applicability of these methods as part of scheduling heuristics, and would like to study such applicability further to consider the potential benefits.

Finally, in the imaging line, we would like to extend the results presented in the thesis. In relation to the good results achieved with ISOMAP and its usefulness with hyperspectral imaging, we want to realize two optimized versions of ISOMAP for GPU and multicore. We have studied the substitution of KNN by other methods with results that apparently give better results (in terms of speed and accuracy), but we need to perform more extensive tests to be able to confirm these results. On the other hand, we will continue to work on optimized quantum circuits that will allow us to study the possible applications of such computation in this field.

Needless to say that we will continue to keep a close eye on the evolution of quantum computing in order to use any innovations that arise to the benefit of our lines, as well as to contribute as much as possible to the development of quantum computing.

Bibliography

[1] Z. Babar, D. Chandra, H.V. Nguyen, P. Botsinis, Dimitrios Alanis, S.X. Ng, and L. Hanzo. “Duality of quantum and classical error correction codes: Design principles and examples”. In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pages 970–1010.

[2] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. Roux, and M. Ouimet. “Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering”. In: *Advances in neural information processing systems* 16 (2003), pages 177–184.

[3] C. Bernhardt. *Quantum computing for everyone*. Mit Press, 2019.

[4] H.R. Bhagyalakshmi and M.K. Venkatesha. “An improved design of a multiplier using reversible logic gates”. In: *International journal of engineering science and technology* 2.8 (2010), pages 3838–3845.

[5] M. Borengasser, W.S. Hungate, and R. Watkins. *Hyperspectral remote sensing: principles and applications*. CRC press, 2007.

[6] I. Borg and P. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[7] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[8] J. Cheng, M. Grossman, and T. McKercher. *Professional CUDA c programming*. John Wiley & Sons, 2014.

[9] N.C. Cruz. “High-Performance Computing for Optimizing the Design and Control of Solar Power Tower Plants”. PhD thesis. Department of Informatics. Univeresity of Almería, Oct. 2019.

[10] S. A Cuccaro, T.G. Draper, S. A Kutin, and D.P. Moulton. “A new quantum ripple-carry addition circuit”. In: *arXiv preprint quant-ph/0410184* (2004).

[11] J. De Leeuw. “Applications of convex analysis to multidimensional scaling”. In: *Recent development in statistics* (2005), pages 133–145.

[12] J. Dhont. *An introduction to dynamics of colloids*. Elsevier, 1996.

[13] T.G. Draper, S.A. Kutin, E.M. Rains, and K.M. Svore. “A logarithmic-depth quantum carry-lookahead adder”. In: *arXiv preprint quant-ph/0406142* (2004).

[14] G. Dzemyda, O. Kurasova, and J. Zilinskas. “Multidimensional data visualization”. In: *Methods and applications series: Springer optimization and its applications* 75 (2013), page 122.

[15] R. Farber. *CUDA application design and development*. Elsevier, 2011.

[16] J.M. Garcia-Martinez, E.M. Garzón, and P.M. Ortigosa. “A GPU implementation of a hybrid evolutionary algorithm: GPuEGO”. In: *The Journal of Supercomputing* 70.2 (2014), pages 684–695.

[17] W. Gropp, W.D. Gropp, E. Lusk, A. Skjellum, and A. Lusk. *Using MPI: portable parallel programming with the message-passing interface*. Volume 1. MIT press, 1999.

[18] L.K. Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pages 212–219.

[19] J.C. Harsanyi and C. Chang. “Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach”. In: *IEEE Transactions on geoscience and remote sensing* 32.4 (1994), pages 779–785.

[20] H. Hasimoto. “On the periodic fundamental solutions of the Stokes equations and their application to viscous flow past a cubic array of spheres”. In: *J. Fluid Mech* 5.02 (1959), pages 317–328.

[21] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot. “Graph convolutional networks for hyperspectral image classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* (2020).

[22] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski. “Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis”. In: *IEEE transactions on Computer-Aided Design of integrated circuits and Systems* 25.9 (2006), pages 1652–1663.

[23] S. Ingram, T. Munzner, and M. Olano. “Glimmer: Multilevel MDS on the GPU”. In: *IEEE Transactions on Visualization and Computer Graphics* 15.2 (2008), pages 249–261.

[24] D.B. Johnson. “A note on Dijkstra’s shortest path algorithm”. In: *Journal of the ACM (JACM)* 20.3 (1973), pages 385–388.

[25] P. Kaur and B.S. Dhaliwal. “Design of fault tolerant full adder/subtractor using reversible gates”. In: *2012 International Conference on Computer Communication and Informatics*. IEEE. 2012, pages 1–5.

[26] P.K. Kumar, P.P. Rao, and K.H. Kishore. “Optimal design of reversible parity preserving new full adder/full subtractor”. In: *2017 11th International Conference on Intelligent Systems and Control (ISCO)*. IEEE. 2017, pages 368–373.

[27] E. Lange, J.B. Caballero, A.M. Puertas, and M. Fuchs. “Comparison of structure and transport properties of concentrated hard and soft sphere fluids”. In: *The Journal of chemical physics* 130.17 (2009), page 174903.

[28] Y. Hui Lee, M. Khalil-Hani, and M.N. Marsono. “An FPGA-based quantum computing emulation framework based on serial-parallel architecture”. In: *International Journal of Reconfigurable Computing* 2016 (2016).

[29] H. Li, P. Fan, H. Xia, H. Peng, and G. Long. “Efficient quantum arithmetic operation circuits for quantum image processing”. In: *SCIENCE CHINA Physics, Mechanics & Astronomy* 63 (2020), pages 1–13.

[30] J. Lobera, G. Ortega, I. García, M.P. Arroyo, and E.M. Garzón. “High performance computing for a 3-D optical diffraction tomographic application in fluid velocimetry”. In: *Optics Express* 23.4 (2015), pages 227–238.

[31] E. Loh. “The ideal HPC programming language”. In: *Communications of the ACM* 53.7 (2010), pages 42–47.

[32] D. Maslov, G.W. Dueck, D.M. Miller, and C. Negrevergne. “Quantum circuit simplification and level compaction”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.3 (2008), pages 436–444.

[33] M. Mastriani. “Quantum image processing?” In: *Quantum Information Processing* 16.1 (2017), pages 1–42.

[34] L.F. Menezes, D.M. Neto, M.C. Oliveira, and J.L. Alves. “Improving Computational Performance through HPC Techniques: case study using DD3IMP in-house code”. In: *AIP Conference Proceedings*. Volume 1353. 1. American Institute of Physics. 2011, pages 1220–1225.

[35] S.K. Mitra and A.R. Chowdhury. “Minimum cost fault tolerant adder circuits in reversible logic synthesis”. In: *2012 25th International Conference on VLSI Design*. IEEE. 2012, pages 334–339.

[36] M. Mohammadi and M. Eshghi. “On figures of merit in reversible and quantum logic designs”. In: *Quantum Information Processing* 8.4 (2009), pages 297–318.

[37] J.M. Molero, E.M. Garzon, I. García, E.S. Quintana-Orti, and A. Plaza. “Efficient implementation of hyperspectral anomaly detection techniques on GPUs and multicore processors”. In: *IEEE Journal of selected topics in applied earth observations and remote sensing* 7.6 (2014), pages 2256–2266.

[38] A.N. Nagamani, S. Ashwin, and V.K. Agrawal. “Design of optimized reversible binary adder/subtractor and BCD adder”. In: *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE. 2014, pages 774–779.

[39] A.N. Nagamani, C. Ramesh, and V.K. Agrawal. “Design of optimized reversible squaring and sum-of-squares units”. In: *Circuits, Systems, and Signal Processing* 37.4 (2018), pages 1753–1776.

[40] B. Nichols, D. Buttlar, and J.P. Farrell. *Pthreads programming*. O’Reilly & Associates, Inc., 1996.

[41] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. New York, NY, USA, 2011.

[42] CUDA Nvidia. *Cublas library user guide*. 2013.

[43] S. Oaks and H. Wong. *Java threads*. O’Reilly Media, Inc., 1999.

[44] G. Oger, D. Le Touzé, D. Guibert, M. De Leffe, J. Biddiscombe, J. Soumagne, and J. Piccinali. “On distributed memory MPI-based parallelization of SPH codes in massive HPC context”. In: *Computer Physics Communications* 200 (2016), pages 1–14.

[45] G. Ortega. “High performance computing for solving large sparse systems. Optical diffraction tomography as a case of study”. PhD thesis. May 2015.

[46] G. Ortega, A. Puertas, and E.M. Garzón. “Accelerating the problem of microrheology in colloidal systems on a GPU”. In: *The Journal of Supercomputing* 73.1 (2017), pages 370–383.

[47] F. Orts, E. Filatovas, G. Ortega, O. Kurasova, and E.M. Garzón. “Improving the energy efficiency of SMACOF for multidimensional scaling on modern architectures”. In: *The Journal of Supercomputing* 75.3 (2019), pages 1038–1050.

[48] F. Orts, G. Ortega, E.F. Combarro, and E.M. Garzón. “A review on reversible quantum adders”. In: *Journal of Network and Computer Applications* (2020), pages 102810–102826.

[49] F. Orts, G. Ortega, A.C. Cucura, E. Filatovas, and E.M. Garzón. “Optimal fault-tolerant quantum comparators for image binarization”. In: *The Journal of Supercomputing* (2021), pages 1–12.

[50] F. Orts, G. Ortega, E. Filatovas, O. Kurasova, and E.M. Garzón. “Hyperspectral image classification using Isomap with SMACOF”. In: *Informatica* 30.2 (2019), pages 349–365.

[51] F. Orts, G. Ortega, and E.M. Garzon. “Efficient reversible quantum design of sign-magnitude to two’s complement converters”. In: *Quantum Information & Computation* 20.9-10 (2020), pages 747–765.

[52] F. Orts, G. Ortega, and E.M. Garzón. “An optimized quantum circuit for converting from sign–magnitude to twoâs complement”. In: *Quantum Information Processing* 18.11 (2019), pages 1–14.

[53] F. Orts, G. Ortega, E.M. Garzón, M. Fuchs, and A.M. Puertas. “Dynamics and friction of a large colloidal particle in a bath of hard spheres: Langevin dynamics simulations and hydrodynamic description”. In: *Physical Review E* 101.5 (2020), page 052607.

[54] F. Orts, G. Ortega, E.M. Garzón, and A.M. Puertas. “Finite size effects in active microrheology in colloids”. In: *Computer Physics Communications* 236 (2019), pages 8–14.

[55] F. Orts, G. Ortega, A.M. Puertas, I. García, and E.M. Garzón. “On solving the unrelated parallel machine scheduling problem: active microrheology as a case study”. In: *The Journal of Supercomputing* 76.11 (2020), pages 8494–8509.

[56] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J.I. Latorre. “Data re-uploading for a universal quantum classifier”. In: *Quantum* 4 (2020), page 226.

[57] C. Pheatt. “Intel® threading building blocks”. In: *Journal of Computing Sciences in Colleges* 23.4 (2008), pages 298–298.

[58] A.M. Puertas and T. Voigtmann. “Microrheology of colloidal systems”. In: *Journal of Physics: Condensed Matter* 26.24 (2014), page 243101.

[59] H.G. Rangaraju, U. Venugopal, K.N. Muralidhara, and K.B. Raja. “Low power reversible parallel binary adder/subtractor”. In: *International journal of VLSI design and Communication Systems* 1.3 (2010), pages 23–34.

[60] F. Rizzo, B. Carpentieri, G. Motta, and J.A. Storer. “Low-complexity lossless compression of hyperspectral imagery via linear prediction”. In: *IEEE Signal Processing Letters* 12.2 (2005), pages 138–141.

[61] R. Saligram and T.R. Rakshith. “Design of low logical cost adders using novel parity conserving Toffoli gate”. In: *2013 International Conference on Emerging Trends in Communication, Control, Signal Processing and Computing Applications (C2SPCA)*. IEEE. 2013, pages 1–6.

[62] R. Sarma and R. Jain. “Quantum gate implementation of a novel reversible half adder and subtractor circuit”. In: *2018 International Conference on Intelligent Circuits and Systems (ICICS)*. IEEE. 2018, pages 72–76.

[63] P.W. Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pages 124–134.

[64] V.P. Singh and M. Rai. “Verilog design of full adder based on reversible gates”. In: *2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall)*. IEEE. 2016, pages 1–5.

[65] N.N. Sirhan. “Multi-core processors: Concepts and implementations”. In: *Available at SSRN 3628131* (2020).

[66] T. Soyata. *GPU parallel program development using CUDA*. CRC Press, 2018.

[67] Y. Takahashi and N. Kunihiro. “A linear-size quantum circuit for addition with no ancillary qubits”. In: *Quantum Information & Computation* 5.6 (2005), pages 440–448.

[68] Y. Takahashi, S. Tani, and N. Kunihiro. “Quantum addition circuits and unbounded fan-out”. In: *arXiv preprint arXiv:0910.2530* (2009).

[69] H. Thapliyal. “Mapping of subtractor and adder-subtractor circuits on reversible quantum gates”. In: *Transactions on Computational Science XXVII*. Springer, 2016, pages 10–34.

[70] H. Thapliyal, H.V. Jayashree, A.N. Nagamani, and H.R. Arabnia. “Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder”. In: *Transactions on Computational Science XVII*. Springer, 2013, pages 73–97.

[71] H. Thapliyal, E. Muñoz-Coreas, and V. Khalus. “T-count and Qubit Optimized Quantum Circuit Designs of Carry Lookahead Adder”. In: *arXiv preprint arXiv:2004.01826* (2020).

[72] H. Thapliyal and N. Ranganathan. “A new reversible design of BCD adder”. In: *2011 Design, Automation & Test in Europe*. IEEE. 2011, pages 1–4.

[73] H. Thapliyal and N. Ranganathan. “Design of efficient reversible logic-based binary and BCD adder circuits”. In: *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 9.3 (2013), pages 1–31.

[74] J. Tompson and K. Schlachter. “An introduction to the opencl programming model”. In: *Person Education* 49 (2012), page 31.

[75] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, and Y. Wang. “Intel math kernel library”. In: *High-Performance Computing on the Intel® Xeon Phiâ¢*. Springer, 2014, pages 167–188.

[76] F. Wang, M. Luo, H. Li, Z. Qu, and X. Wang. “Improved quantum ripple-carry addition circuit”. In: *Science China Information Sciences* 59.4 (2016), page 042406.

[77] D. Whitley. “A genetic algorithm tutorial”. In: *Statistics and computing* 4.2 (1994), pages 65–85.

[78] F. Wickelmaier. “An introduction to MDS”. In: *Sound Quality Research Unit, Aalborg University, Denmark* 46.5 (2003), pages 1–26.

[79] H. Xia, H. Li, H. Zhang, Y. Liang, and J. Xin. “An efficient design of reversible multi-bit quantum comparator via only a single ancillary bit”. In: *International Journal of Theoretical Physics* 57.12 (2018), pages 3727–3744.

[80] H. Xia, H. Li, H. Zhang, Y. Liang, and J. Xin. “Novel multi-bit quantum comparators and their application in image binarization”. In: *Quantum Information Processing* 18.7 (2019), pages 1–17.

[81] S. Yamashita, S. Minato, and D.M. Miller. “DDMF: An efficient decision diagram structure for design verification of quantum circuits under a practical restriction”. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 91.12 (2008), pages 3793–3802.

[82] M. Yao and B. Vocational. “Research on learning evidence improvement for KNN based classification algorithm”. In: *Int. J. Database Theory Appl* 7.1 (2014), pages 103–110.

[83] Y. Zhang, K. Lu, Y. Gao, and M. Wang. “NEQR: a novel enhanced quantum representation of digital images”. In: *Quantum information processing* 12.8 (2013), pages 2833–2860.

[84] R. Zhou, Y. Li, and M. Zhang. “Novel designs for fault tolerant reversible binary coded decimal adders”. In: *International Journal of Electronics* 101.10 (2014), pages 1336–1356.