POSGRADO EN INFORMÁTICA

# PROYECTO FIN DE MÁSTER

## UNIVERSIDAD DE ALMERIA

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN TÉCNICAS INFORMÁTICA AVANZADAS

# Optimización de problemas multi-objetivo de empaquetado de palets mediante algoritmos evolutivos


**Curso 2010/2011**

**Alumno/a:**
María Parra Amat

**Director/es:**
Consolación Gil Montoya

# Optimización de problemas multi-objetivo de empaquetado de palets mediante algoritmos evolutivos

María Parra Amat
Dpt. Arquitectura de Computadores y Electrónica, Universidad de Almería
Carretera de Sacramento s/n, La Cañada de San Urbano, 04120
Almería (España) (+34) 630341318
mariaparra@ual.es

## ABSTRACT

A parallel evolutionary multi-objective algorithm has been implemented to solve the problem of packing in two dimensions with restrictions. This algorithm has been used to solve the real-word application of palets transport in trucks. The transport of palets in trucks has a great importance in Andalusia and especially in Almería's field where hundreds of trucks go out towards Europe loaded with all kind of products daily. The problem of packing in two dimensions (2DPP) consists of packing a collection of objects, characterized by having different heights and widths, in the minimum number of bins (containers). For this problem there exists multitude of variants. In the multi-objective variant of the problem, beside minimizing the number of containers, is minimized trying to place the load of the pieces as best as possible in order that the gravity center of the container remains as near as possible to the desired center. The algorithm has been applied to a variant of the problem where a set of palets with his high place, width and specific weight, have to be inserted in the minimum in the minor possible number of trucks and with the best balancing of load, avoiding a series of restrictions added to the problem. Every palet corresponds to a client, where all the palets of the same client must go in the same truck, every truck cannot be loaded by more than 25000 kilos and the gravity center must go near the axis of the truck. For the optimization of this problem we have implemented an evolutionary algorithm, which, is inspired in the theory of Darwin's evolution and in the development of the evolutionary computer science. The evolutionary algorithms are technologies of optimization and search of solutions based on the natural and genetic selection that allow to solve non-linear problems in which a high number of variables in complex problems are controled. The algorithm has been implemented by a set of evolutionary operators designed to obtain solutions of great quality for a set of established instances.

## Keywords

Packing problems, evolutionary algorithms, optimization multi-objective, parallel optimization and heuristics, palets transport.

## RESUMEN

En este trabajo se ha implementado un algoritmo evolutivo multi-objetivo paralelo, para resolver el problema de empaquetamiento en dos dimensiones con restricciones, para una aplicación de transporte de palets en camiones. El transporte de palets en camiones tiene una gran importancia en Andalucía y especialmente en el campo almeriense donde a diario salen hacia Europa cientos de camiones cargados de productos del campo. El problema de empaquetamiento en dos dimensiones (2DPP) consiste en insertar un conjunto de objetos caracterizados por tener un alto y ancho específico, en el menor número de camiones posibles donde el alto y ancho es igual para todos. A partir de esta definición existen multitud de variantes al problema. En la variante multi-objetivo del problema, además de minimizar el número de camiones, se intenta minimizar el balanceo de carga de los mismos intentando colocar la carga de las piezas de la mejor forma posible para que el centro de gravedad del camión quede lo más cercano posible al centro deseado. El algoritmo se ha aplicado a una variante del problema donde se trata de insertar un conjunto de palets con su alto, ancho y peso específico, en el menor número de camiones posible y con el mejor balanceo de carga, evitando una serie de restricciones añadidas al problema. Cada palet corresponde a un cliente, con lo cual todos los palets de un mismo cliente deben de ir en el mismo camión, cada camión no puede ir cargado con más de 25000 kilos y el centro de gravedad debe de ir lo más próximo al eje del camión. Para la optimización de este problema hemos implementado un algoritmo evolutivo TP-MOEA, este tipo de algoritmos están inspirados en la teoría de la evolución de Darwin y en el desarrollo de la informática evolutiva. Los algoritmos evolutivos son técnicas de optimización y búsqueda de soluciones basadas en la selección natural y genética que permiten resolver problemas no lineales en los que interviene un alto número de variables en problemas complejos. El algoritmo ha sido implementado con un conjunto de operadores evolutivos diseñados para obtener soluciones de gran calidad para un conjunto de instancias establecidas.

## Palabras clave

Problemas de empaquetamiento, algoritmos evolutivos, optimización multi-objetivo, optimización paralela, heurísticas, transporte de palets.

# 1. INTRODUCCIÓN

En muchas aplicaciones industriales es necesario asignar un conjunto de elementos rectangulares en grandes superficies rectangulares de un tamaño estándar, tratando de aprovechar al máximo los espacios libres. En las industrias de la madera o el vidrio, estos componentes rectangulares tienen que ser cortados en grandes hojas de material. En el contexto del almacenamiento, las mercancías tienen que ser colocadas en los distintos estantes para su almacenamiento. En los periódicos la paginación y la asignación de las columnas entre los distintos artículos y anuncios tienen que estar dispuestos de la forma más eficiente en las distintas páginas. En estos casos mencionados, las unidades que se almacenan son rectángulos, y la función objetivo a optimizar es empaquetar todas estas unidades en el menor número de almacenes posibles. Este tipo de problemas de empaquetamiento es conocido en la literatura como problema de empaquetamiento en dos dimensiones. En otros contextos, tales como las industrias de papel o tela, tenemos una sola unidad en la que insertar las piezas rectangulares (un rollo de material), y el objetivo es obtener las piezas mediante el uso de la longitud del rollo mínimo: este problema es denominado como empaquetamiento en tira (strip packing) en dos dimensiones. La mayoría de las contribuciones en la literatura tratan el problema con piezas en las que la orientación es fija sin poder rotarlas a la hora de empaquetarlas. En Dyckhoff y Finke [1], Dowsland [2] y Dyckhoff y col. [3] se realiza un estado del arte sobre los problemas de empaquetamiento y corte mencionados anteriormente. En Coffman y Shor [4], Coffman y Lueker [5] se realiza un análisis probabilístico de los resultados obtenidos por diferentes algoritmos para problemas de tipo empaquetamiento.

Para el problema de empaquetamiento en dos dimensiones (2DPP), se parte de un conjunto de contenedores de número ilimitado iguales con un ancho $W$ y un alto $H$, donde el objetivo es insertar todas las piezas en el menor número de contenedores posibles.

Para el caso del problema de strip packing en dos dimensiones (2SPP), se tiene un único contenedor con ancho $W$, y una altura infinita (llamada tira) donde el objetivo es minimizar la altura de la tira a la hora de insertar todas las piezas.

En ambos casos, las piezas tienen que ser insertadas con las aristas paralelas a los lados de los contenedores. Ambos problemas son de tipo NP-duro, es decir, no existe método conocido que pueda resolverlos en un tiempo polinomial, razón por la cual es aconsejable aplicar métodos aproximados.

En el caso del problema de empaquetamiento, algunos autores como Liu y Tan [6] han añadido un nuevo objetivo al problema que consiste en minimizar el balanceo de carga de los contenedores, convirtiendo el problema en multi-objetivo.

Un tercer caso relevante de empaquetamiento de rectángulos es el siguiente: un conjunto de piezas tienen un beneficio $p_i>0$, y el problema consiste en seleccionar un conjunto de piezas para insertar en un único contenedor de tamaño fijo, de tal forma que se maximice el beneficio total de las piezas insertadas. Este problema se denomina Cutting Stock en dos dimensiones y fue introducido por Gilmore y Gomory [7] como cutting knapsack en dos dimensiones.

En este trabajo nos vamos a centrar en resolver el problema de empaquetamiento en dos dimensiones para una aplicación real de almacenamiento de palets en diferentes camiones. Cuando analizamos el transporte de palets en camiones nos encontramos con una serie de restricciones importantes a la hora de realizar esta tarea. Una de ellas consiste en no superar un peso máximo en el remolque a la hora de cargarlo, así como distribuir el peso de este de forma que quede la mayor parte del peso en el eje del camión para una mayor seguridad en el transporte. En cuanto a la logística no siempre un camión va a un solo cliente como destino final, sino que suele llevar palets para diferentes clientes, además los palets de un mismo cliente deben de ir en un mismo camión para tener una mayor eficiencia en el transporte. Con estas restricciones se ha implementado un algoritmo evolutivo multi-objetivo para optimizar el problema de empaquetamiento de palets en dos dimensiones con restricciones llamado TP-MOEA.

En cuanto a la organización del documento, indicar que en la sección 2 se ofrece una visión general sobre el sector transportes en el que se detalla la importancia económica del mismo así como una clasificación de los distintos problemas que han ido surgiendo a lo largo del tiempo. La sección 3 describe con detalle el problema de empaquetamiento de palets en dos dimensiones multi-objetivo. La Sección 4 presenta la estructura del algoritmo evolutivo que se ha implementado, así como el funcionamiento de los operadores evolutivos. La sección 5 muestra los resultados obtenidos de ejecutar el algoritmo evolutivo para un conjunto de instancias predefinidas. Finalmente, la sección 6 ofrece las conclusiones del trabajo y describen algunas líneas de trabajo futuro.

# 2. EL SECTOR TRANSPORTES

## 2.1 La importancia del sector

En España, el transporte es un sector económico de una importancia estratégica creciente, no sólo por contribuir a la mejora de la competitividad de nuestro país, sino por apoyar el desarrollo de la actividad en otros sectores como la industria, el comercio y el turismo, por citar aquellos con mayor peso en el tejido productivo de la economía española. Esta relevancia aumenta si se tiene en cuenta el proceso de globalización, que exige mayor capacidad para atender el volumen creciente de intercambios comerciales y de pasajeros a escala mundial. La mayor apertura de la economía junto con la mayor competencia internacional lleva implícito un sistema de suministro más flexible, fiable y rápido, que necesita el desarrollo de tecnología puntera para satisfacer una demanda cada día más especializada. Por lo tanto, este sector es fuente de innovación. La posición geográfica de España, como puente entre Europa, América Latina y África, imprime un carácter especialmente destacado a los servicios de transporte y aumenta el potencial de crecimiento

de su actividad. La importancia del sector de transportes queda reflejada en su peso en el tejido productivo. En términos de Valor Añadido Bruto, la contribución del sector del transporte a la riqueza nacional se consolida en el entorno del 5%, con leves variaciones en el periodo comprendido entre 2000 y 2005. Por subsectores, el transporte terrestre concentra aproximadamente algo más de la mitad del VAB del sector transportes en 2005, seguido de un 8% del transporte aéreo y de un 2,6% del transporte marítimo.


**Figura 1.** La importancia del sector transportes en España.

En cuanto al número de empresas, el sector cuenta en 2008 con 238.119 empresas, lo que supone el 7% del total de empresas en España, según información obtenida a partir del DIRCE [8]. Por subsectores, en los últimos años (2000-2008), y sin perjuicio del crecimiento dentro de las actividades conexas al Sector, el mayor crecimiento ha tenido lugar en el transporte aéreo (50%), seguido del transporte marítimo (16,8%) y, finalmente, el transporte terrestre (4,8%).

Con la información disponible hasta 2007, estas empresas generan un volumen de negocio de 98552,1 millones de euros, lo que supone un crecimiento del 73% respecto al año 2000. Por su parte, cabe señalar que este sector ha realizado un notable esfuerzo inversor. La inversión fue de 13154,7 millones de euros en 2007, que más que duplica la realizada a comienzos de la década actual. Por lo tanto, el transporte ha contribuido de manera destacada a elevar el stock de capital en nuestro país y, con ello, a incrementar el crecimiento potencial de la economía española.

En España a fecha de 2008 había registradas 45.710 empresas relacionadas con el transporte de mercancías por carretera, cifra que indica la importancia de este sector en la economía de este país. España es el primer país exportador de frutas y el segundo de hortalizas frescas por detrás de Italia a nivel europeo. En el caso de las hortalizas el 95% de las exportaciones se dirigen al mercado europeo. La exportación de frutas alcanzó en el 2008 la cifra de 5,4 millones de toneladas y de 4 millones en el caso de las hortalizas. Por mercados de destino, las ventas exteriores de frutas y hortalizas frescas se dirigen mayoritariamente a la Unión Europea, siendo los principales mercados Alemania, Reino Unido, Francia y Países Bajos. En el caso de la Economía de Almería representa el 1.2% del total nacional. Un porcentaje que la sitúa en el puesto 25 del ranking provincial. El sector agroindustrial representa, de forma directa, el 23% de la economía de Almería, frente al 5% que supone en el total de la actividad económica española.

## 2.2 Situación del transporte en España

El transporte de mercancías por carretera se ha incrementado desde 719.335 toneladas (690.807 nacionales y 28.528 internacionales) (importaciones/exportaciones) hasta las 1.541.653 (1.474.552 nacionales y 67.101 internacionales) en 2008 (Ministerio de Fomento, 2010).

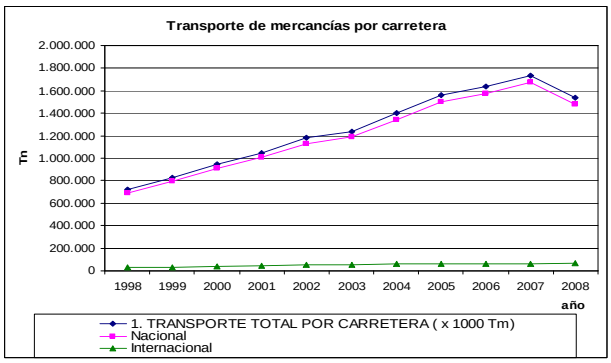


**Figura 2.** Transporte por carretera. Elaboración propia a partir de los datos del Ministerio de Fomento.

El transporte de mercancías movió unas 31 millones de toneladas. La figura 2 muestra, pese al estancamiento o ligera reducción del último año debido a la coyuntura económica, una tendencia de largo plazo creciente en el uso del transporte por carretera. La mejora de las infraestructuras de transporte en nuestro país se han visto impulsadas por el esfuerzo inversor que, durante las últimas décadas, han llevado a cabo tanto los sucesivos gobiernos nacionales, autonómicos, y provinciales, como la Unión Europea a través de los fondos regionales y de cohesión. En particular, y a nivel nacional, el Ministerio de Fomento, a través de sucesivas etapas ha llevado a cabo diferentes planes para la mejora del transporte, lo que ha conllevado a su vez una gran inversión económica en obras públicas relacionadas con infraestructuras de transporte. En este sentido se puede destacar el Plan Director de Infraestructuras 1993-2007, con un presupuesto de 112.390 millones de euros, el Plan de Infraestructuras 2000-2007, con un presupuesto de 192.323 millones de euros, y el actual Plan Estratégico de Infraestructuras y Transporte 2005-2020, el cual espera desarrollar inversiones por valor de unos 250.000 millones de euros.

## 2.3 Situación del transporte en la Unión Europea. Redes transeuropeas de transporte.

El transporte es un componente esencial de la economía europea. La industria del transporte repercute cerca de 7% del producto interno bruto (PIB) y más del 5% del empleo total en la UE. La mejora y la promoción de modos de transporte alternativos a la carretera, tanto para la movilidad de pasajeros y de carga, es una de las prioridades de la política de la Unión Europea para desarrollar un sistema sostenible de transporte. La

Política Europea de Transporte (PET) tiene por objeto establecer un sistema de transporte sostenible que satisfaga las necesidades económicas de la sociedad, las necesidades sociales y ambientales y es conducente a una sociedad inclusiva y una Europa plenamente integrada y competitiva. Las prioridades más inmediatas parecen ser la mejor integración de los diferentes modos de transporte como una forma de mejorar la eficacia global del sistema y la aceleración del desarrollo y despliegue de tecnologías innovadoras.

El transporte de mercaderías aumentó en la UE, en promedio, un 2,7% por año. Cabe destacar, el crecimiento del transporte de mercancías también está vinculado a prácticas económicas tales como la concentración de la producción en un número reducido de lugares (deslocalización) aprovechando las economías de escala, etc.

Otro foco de interés importante de la PET consiste en aumentar la eficiencia energética del transporte, pero no se ha visto refrendado por una reducción del consumo global de combustibles. De hecho, el sector del transporte es responsable de una cantidad significativa de las emisiones de efecto invernadero: el 13% de todas las emisiones de gases de efecto invernadero y el 23% de las emisiones mundiales de CO2 procedentes de la combustión de combustibles fósiles [9], suponiendo el transporte de mercancías un tercio de las emisiones de CO2 del sector del transporte mundial en su conjunto [9]. La gran demanda de transporte por carretera, que aún se espera que aumente en los próximos años, genera importantes externalidades negativas en términos de costos ambientales y sociales (congestión, contaminación, daños de infraestructura, los accidentes de tráfico), y aumenta el uso excesivo de las redes viarias. El desarrollo de los sistemas de transporte intermodal es una solución alternativa para moderar el aumento de las externalidades negativas mediante la sustitución de menos recursos que consumen los modos de transporte por carretera. En este sentido, las infraestructuras necesitan ser cuidadosamente planificadas y priorizadas con miras a la optimización de las cadenas de transporte y la red de transporte en general. Además de la eliminación de los cuellos de botella, será esencial para identificar a los corredores verdes con el fin de reducir la congestión y la contaminación ambiental. Se espera que las alternativas al transporte de carretera permitan absorber parte del futuro incremento en la demanda de transporte. En particular, con respecto al transporte de pasajeros, la integración del transporte aéreo y del ferrocarril de alta velocidad será un avance crucial. En cuanto al transporte de mercancías, se deben fomentar sistemas logísticos inteligente e integrados, donde el desarrollo de los puertos y terminales intermodales, se convierta en un elemento clave.

La ciencia y la industria ya son muy activas en la búsqueda de soluciones para la seguridad del transporte, la dependencia de los combustibles, emisiones de vehículos y la congestión de la red, lo cual es especialmente importante considerando las previsiones de la Unión Europea para las próximas décadas.

## 2.4 Clasificación de los problemas de transporte

Los problemas de transporte suelen ser clasificados en tres categorías diferentes [10,11,12]:

- *Planificación estratégica (largo plazo):* a nivel organizativo, implica típicamente el más alto nivel de gestión y requiere grandes inversiones de capital en horizontes a largo plazo. Estas incluyen el diseño de la red física y de su evolución, la ubicación de las instalaciones principales como son los terminales), etc. Así, la planificación estratégica puede ser dividida en dos subcategorías principales:

  a) *Planificación estratégica en redes. Diseño del sistema:* este nivel estratégico permite tomar una decisión sobre la ubicación de terminales (modelos de ubicación) y el diseño global de la red (modelos de diseño de red). Estos problemas, que son NP-completos [13], suelen ser modelados por formulaciones estáticas y deterministas. Dentro de este tipo de problemas encontramos el diseño de la red (diseño de las carreteras, líneas de ferrocarril, o las autopistas virtuales aéreas o del mar), y los problemas de localización, que consisten en determinar la posición donde situar nodos terminales o de enlace (estaciones, aeropuertos, puertos, etc.)

  *b) Planificación estratégica en redes. Suministro, demanda y asignación:* esos estudios suelen usar análisis de coste-beneficio, evaluación y comparación de alternativas de inversión, y son generalmente requeridas por agencias nacionales o regionales.

- *Planificación táctica (medio plazo):* esta planificación aspira a determinar, en un horizonte temporal medio, la localización y utilización eficiente de los recursos para alcanzar el mayor rendimiento posible para el problema en conjunto. Las principales decisiones a tomar en este nivel hacen referencia al diseño del servicio de red, incluyendo la selección de servicio (frecuencia y planificación de decisiones), distribución del tráfico (itinerarios y rutas) y políticas en terminales (actividades de consolidación en terminales intermodales). Dichos problemas suelen clasificarse en dinámicos o estáticos, en función de si se toma en consideración la dimensión temporal.

- *Planificación operativa (corto plazo):* esta planificación se lleva a cabo a nivel local, y las principales decisiones asociadas tienen que ver con aspectos de planificación y gestión como la planificación del trabajo de empleados, planificación de operaciones de mantenimiento, consignar vehículos, etc. Este tipo de problemas aparecen, principalmente, en el transporte de mercancías, siendo el almacenamiento de contenedores uno de los más tratados.

El problema de empaquetamiento de palets en camiones, con el que vamos a trabajar, se encuadra en los problemas de planificación operativa.

## 2.5 El transporte de palets en camiones

Almería es un sitio estratégico en el transporte de mercancías a Europa por su intensa actividad agrícola, esto conlleva que a diario se carguen cientos de trailers con palets cargados de verdura para su transporte a diversos destinos nacionales e internacionales. Dada la importancia del transporte de mercancías por carretera en esta provincia, vamos a detallar unos de los problemas con los que se trata a diario en este sector. En Europa el palet más utilizado es el palet europeo (Figura 3), debido a sus medidas (1200 por 800 milímetros), es idóneo a la hora de colocarlo en los remolques cuyas medidas

estándares rondan los 2,4 metros de ancho por 13,5 metros de profundidad por 2,6 de alto. Las figuras 4 y 5 muestran cómo se pueden introducir los palets de dos formas para conseguir introducir 33 palets en el contenedor de un trailer con medidas estándares.

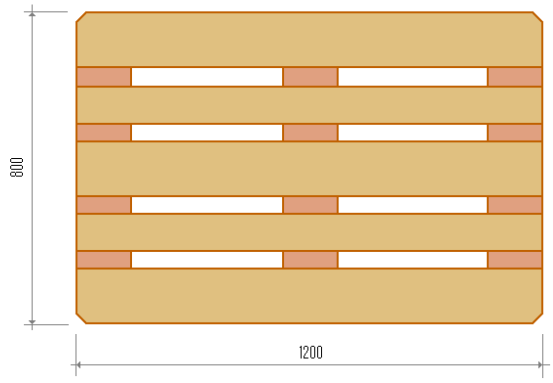


**Figura 3.** Palet europeo de 1200x800 mm.

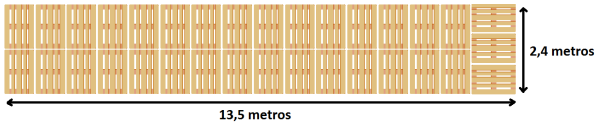


**Figura 4.** Configuración de los palets en estampa.

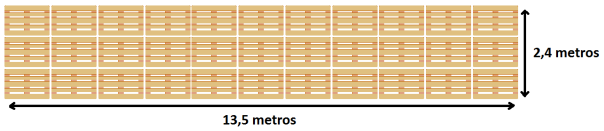


**Figura 5.** Configuración de los palets en fila.

## 2.6 La importancia del balanceo de carga

Cada camión tiene su capacidad bruta de carga establecida en su proyecto y registrada legalmente. Esta carga bruta se compone del peso de la carrocería y del peso de la carga útil sobre ella transportada. Para llevar a cabo los cálculos de la distribución del peso de la carga bruta entre los ejes del camión, es necesario el concepto de Centro de Gravedad (CG) el cual es el punto donde se aplica una fuerza única que representa la concentración de toda la masa de un determinado cuerpo.

El emplazamiento correcto del GC es muy importante para obtener un mejor rendimiento, desempeño y vida útil del camión, por el contrario si el emplazamiento es incorrecto las consecuencias serían una carga inferior a lo posible, menor seguridad respecto a accidentes, velocidad media inferior, mayores gastos de mantenimiento, menor vida útil del camión, mayor consumo de combustible.

La regulación actual limita la carga de los camiones por ejes como se muestra en la figura 6. Por ello nuestro algoritmo intentara que el centro de gravedad de la carga quede lo más próximo al centro de gravedad deseado.

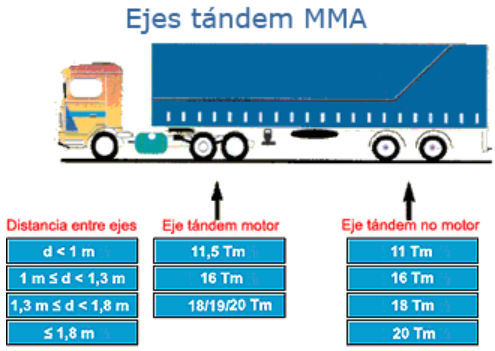


**Figura 6.** Distribución del peso por ejes en un trailer.

Es por esto que es importante la colocación de los palets en distintas posiciones dependiendo del peso de cada palet. En el caso de la agricultura no será lo mismo el peso de un palet de tomates que el de pimientos cuyo peso es inferior.

En este trabajo se ha considerado que para garantizar la distribución adecuada del peso y un correcto emplazamiento del CG, la carga del camión debe recaer sobre los neumáticos delanteros debido a que si los ejes delanteros no tienen suficiente peso de carga (lo que sucede cuando el peso se coloca demasiado hacia atrás) se puede aligerar tanto el peso en el eje de dirección que girar no sea seguro y exista un mayor riesgo de volcar.

# 3. EL PROBLEMA DE EMPAQUETAMIENTO

El problema de empaquetamiento (PP) y sus múltiples variantes tienen un gran número de aplicaciones prácticas en ciencias de la computación (por ejemplo, la asignación de segmentos de pista en discos), en la industria (cortes de metales), en la programación de las máquinas (minimizando el número de máquinas necesarias para completar todas las tareas en un plazo determinado), el transporte y logística (la carga de contenedores, la carga de palets), etc [14]. El problema de empaquetamiento en dos dimensiones (2DPP) [15] consiste en insertar un conjunto de objetos, que se caracteriza por tener diferentes alturas y anchuras, en el menor número de contenedores posibles. La familia de los problemas de empaquetamiento es uno de los problemas de programación considerado NP-duro [13], lo que implica que no se conoce ningún método determinista para la obtención de la solución óptima en un tiempo polinomial. Los problemas de empaquetamiento tienen un gran número de variantes, tales como problemas de corte, cuya principal objetivo es cortar las hojas de metal para la obtención de las piezas con el menor desperdicio posible. Ambos problemas han sido a menudo analizados conjuntamente, y se han referido como cutting and packing problems. Estos problemas poseen diferentes características: el número de dimensiones (piezas con uno, dos o tres dimensiones), la orientación y la forma (regular o irregular) de las piezas, etc.

Pocos autores han tratado el problema de empaquetamiento en dos dimensiones de forma multi-objetivo (MO-2DPP) [16,17] considerando un nuevo objetivo a optimizar. Hasta ahora, el

número de contenedores había sido el único objetivo que se había tratado de optimizar, hasta que recientemente se ha considerado como un segundo objetivo mejorar el balanceo de carga de los contenedores, intentado que el peso de la carga estuviera lo más próximo a un centro de gravedad deseado. Este trabajo analiza el uso de algoritmos evolutivos en la resolución de problemas de empaquetamiento en dos dimensiones multi-objetivo.

## 3.1 Problema de empaquetamiento bi-dimensional (2DPP) multi-objetivo

Pocos trabajos han tratado con el 2DPP para varios objetivos, intentando tan sólo minimizar el número de contenedores posibles. En este sentido, Liu y Tan [6] desarrollaron un algoritmo evolutivo basado en optimización por cúmulo de partículas para el 2DPP multi-objetivo (MO2DPP) añadiendo un nuevo objetivo al problema, el cual consiste en minimizar el balanceo de carga de los contenedores. La optimización multi-objetivo que hemos desarrollado se basa en el frente de dominancia de Pareto para seleccionar las mejores soluciones que nos devuelve el algoritmo.

El MO2DPP puede describirse como:

Dado un conjunto de $n$ piezas rectangulares (objetos) donde $h_i$, $w_i$, y $\gamma_i$ son la altura, ancho y peso del objeto $i$, respectivamente ($i$=1,2,…,$n$), y dado un número ilimitado de contenedores, los cuales tienen una altura $H$, ancho $W$ y centro de gravedad ($\lambda_x$,$\lambda_y$), el objetivo es insertar todas las piezas sin solapamiento en el menor número de contenedores posibles (*nBIN*), con el centro de gravedad (CG) de los contenedores lo más cercano posible al centro deseado CG. El centro deseado CG es la parte inferior del contenedor, en este sentido trata de minimizar la distancia $d_i$ entre el CG del contenedor, alterado por las piezas que contiene, con el centro deseado CG. La Figura 9 muestra un ejemplo del problema.

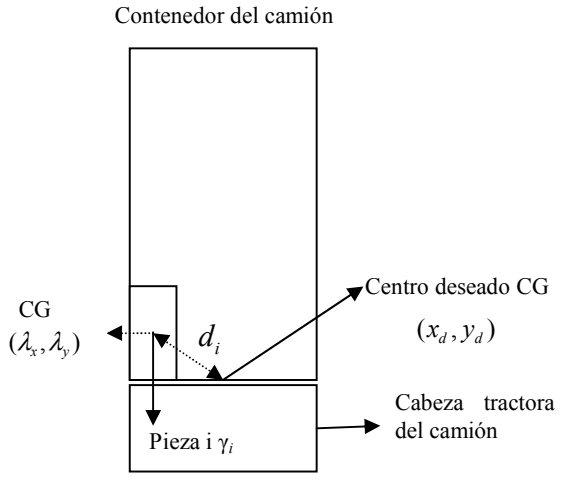


**Figura 9.** Representación gráfica del balanceo de carga.

La definición de los parámetros, los objetivos y restricciones se describen como:

$X_{ij} \in \{0,1\}$, donde $i$=$\{1…N\}$, $j$=$\{1…nBIN\}$. Si la pieza $i$ es asignada dentro del contenedor $j$, $X_{ij}$=1, de lo contrario $X_{ij}$=0;

$N$ Número de piezas;

*nBIN* Número de contenedores;

El centro de gravedad is definido como:

$$CG = \frac{1}{nBIN} \sum_{j=1}^{nBIN} \sqrt{(\lambda_{x,j} - \lambda_{d,x})^2 + (\lambda_{y,j})^2} \quad (1)$$

$$\lambda_{x,j} = \frac{\sum_{i=1}^{n} X_{ij} x_i \gamma_i}{\sum_{i=1}^{n} \gamma_i} \qquad \lambda_{y,j} = \frac{\sum_{i=1}^{n} X_{ij} y_i \gamma_i}{\sum_{i=1}^{n} \gamma_i} \quad (2)$$

donde:

$h_i$ , $w_i$ y $\gamma_i$: altura, ancho y peso de la pieza $i$;

$x_i$ y $\gamma_i$: centro de gravedad de la pieza $i$ en la posición $x$ y $y$;

$H$ y $W$: altura y ancho de los contenedores;

($\lambda_{x,j}$ ,$\lambda_{y,j}$ ): coordenadas del centro de gravedad del contenedor $j$;

$\lambda_{d,x}$: centro de gravedad del contenedor $i$ en la dirección $x$;

$CG$: balanceo de los contenedores acorde al centro de gravedad;

Las restricciones se define como:

Para cada pieza, $i \in \{1,...,N\}$,

$0 < w_i \leq W$;

$0 < h_i \leq H$;

$\gamma_i < 0$;

Para todos los contenedores, $j \in \{1,...,nBIN\}$,

$\sum_{i=1}^{n} X_{ij} h_i w_i \leq WH$

$0 \leq \lambda_x \leq W$

$0 \leq \lambda_y \leq H$

Cada pieza debe ser asignada a un solo contenedor:

$$\sum_{j=1}^{nBIN} \sum_{i=1}^{N} X_{ij} = N$$

Para cada pieza, $i \in \{1,\ldots, N\}$, al ser rotada 90º,

$w_i \leftarrow h_i$ ; $h_i \leftarrow w_i$ ; si $w_i \leq W$ y $h_i \leq H$.

Por lo tanto, para evaluar el balanceo de carga de un individuo la función de aptitud (fitness) calcula el balanceo medio entre todos los contenedores, calculando para cada uno las distancias euclídeas de cada pieza con el centro del contenedor, teniendo en cuenta el peso de cada pieza. El balanceo de carga es una importante aplicación en muchos problemas como carga de contenedores, carga de trailer o carga de pallets en camiones o aviones.

En la siguiente sección se darán los conceptos fundamentales relacionados con las técnicas de optimización estudiadas.

# 4. TÉCNICAS DE OPTIMIZACIÓN

## 4.1 Optimización mono-objetivo

Un problema de optimización consiste en encontrar una combinación óptima de valores para un conjunto de variables, $X=(X_1, X_2,...X_n)$, tal que se maximice o minimice una determinada función, denominada *función objetivo, f(X),* que obviamente es dependiente de dichas variables. Dado que se trata de optimizar una única función objetivo, se habla de problema de optimización mono-objetivo [18].

Pese a la aparentemente simplicidad de la descripción previa, la gran mayoría de problemas de optimización del mundo real, especialmente aquellos de ámbitos científicos, económicos e industriales, requieren de la optimización de funciones objetivo cuyas funciones objetivo a optimizar son extremadamente complejas debido a diferentes motivos como, por ejemplo, que el número de variables sea muy elevado, que los valores que puedan tomar dichas variables sea muy amplio, que la evaluación de la función objetivo sea lenta y compleja, o una combinación de dichas situaciones. Así, bajo estas circunstancias y para problemas de cierto tamaño, resulta inviable encontrar la solución óptima debido a la imposibilidad de evaluar todas las posibilidades en un tiempo acotado. Afortunadamente, el estado actual de la ciencia ayuda a obtener soluciones de gran calidad, no sólo gracias a los modelos matemáticos que se pueden aplicar a estos problemas, principalmente dentro del ámbito de la Investigación Operativa, sino también gracias al gran potencial que ofrece la computación en la actualidad. Pese a ello, la alta complejidad de los problemas de empaquetamiento hace conveniente aplicar técnicas computacionales basadas en meta-heurísticas. A continuación vamos a comentar brevemente algunas de las técnicas heurísticas de optimización más utilizadas [19].

### *4.1.1 Ascenso de colinas (HC, Hill Climbing)*

El ascenso de colinas [20] es la técnica básica de búsqueda local y consiste en tomar una primera solución, generalmente elegida al azar, y realizar variaciones sobre la misma, de forma que si dichas variaciones llevan a una mejora, se conservan los cambios realizados, mientras que en caso contrario son descartados. El proceso finaliza cuando no hay mejora en la solución, razón por la cual esta técnica siempre se detiene en el primer óptimo local que encuentra, lo que no es muy efectivo en problemas combinatorios complejos donde existen múltiples óptimos locales y uno global. Para entender de dónde viene el nombre de esta técnica, imagínese que el espacio de todas las soluciones posibles a un cierto problema se representan como un paisaje tridimensional. Una solución al problema es un conjunto de coordenadas en ese paisaje. Aquéllas que son mejores se encuentran a mayor altitud, formando colinas y picos, mientras que las peores están a menor altura, formando valles. Un algoritmo de ascenso de colinas es, por tanto, un algoritmo que empieza en un punto dado del espacio de búsqueda y admite únicamente soluciones vecinas cuya función objetivo mejore con respecto a la solución previa. Sin embargo, una vez encontrado el óptimo local detiene su búsqueda, por lo que no siempre resulta posible alcanzar el óptimo global.

### *4.1.2 Enfriamiento simulado (SA, Simulated Annealing)*

El enfriamiento o recocido simulado [21] es otra técnica de búsqueda local que intenta evitar el gran inconveniente de los algoritmos de ascenso de colinas, esto es, que se detienen al encontrar el primer óptimo local. Su funcionamiento imita al proceso industrial mediante el cual un material se calienta por encima de su punto de fusión y luego se enfría gradualmente para eliminar defectos en su estructura cristalina, obteniendo así un entramado de átomos más estable y regular. El enfriamiento simulado no trabaja con poblaciones, sino que usa una única solución durante el proceso de optimización. Esta técnica incorpora el concepto de temperatura ($t$), que es una cantidad numérica que se inicializa a un valor $Ti$ ($t = Ti$) y se reduce gradualmente en el tiempo dependiendo de cierto factor de enfriamiento (*Tenfr*). En cada iteración del algoritmo la solución cambia. Si mejora la aptitud, se acepta la nueva solución. Si no, la probabilidad de ser aceptada dependerá de cómo de alta sea la temperatura (a mayor temperatura, mayor probabilidad). El algoritmo termina cuando la temperatura cae por debajo de un determinado umbral (*Tparada*).

La técnica comienza asignando a la temperatura un valor alto. Esto proporciona una probabilidad también alta de aceptar un movimiento de no mejora, con el fin de escapar de óptimos locales. En cada iteración se va reduciendo la temperatura (normalmente se multiplica *Tenfr* por $t$) y, por lo tanto, disminuye la probabilidad de aceptar peores soluciones. Al principio, la reducción de la temperatura es mucho más acusada, mientras que cuando se alcanza una temperatura relativamente baja, la velocidad de reducción se suaviza, tal y como ocurre en el proceso físico real al que se imita. La elección de *Ti, Tenfr* y *Tparada* es primordial si queremos que el enfriamiento simulado tenga éxito en su búsqueda de la solución óptima. Trabajos anteriores [22] han demostrado que esquemas de enfriamiento lentos (valores de *Tenfr* próximos a la unidad) funcionan correctamente. No existe unanimidad en cuanto al número de iteraciones que deben realizarse con cada temperatura, pero en lo que si están de acuerdo todos los autores es que es mucho mejor invertir más tiempo en temperaturas bajas para asegurar que el espacio de búsqueda haya sido explorado en profundidad. En teoría, el algoritmo debería terminar cuando *Tparada* fuese 0. Sin embargo, no es necesario llegar a dicho valor, sino a un umbral próximo a él.

### *4.1.3 Búsqueda Tabú (TS, Tabu Search)*

Tiene sus orígenes en diversos trabajos publicados a principios de la década de los ochenta, aunque fue presentada formalmente por Fred Glover y otros investigadores años más tarde [23]. Se trata de una técnica meta-heurística que puede ser acoplada a cualquier procedimiento de búsqueda local con la intención de guiarlo evitando que quede atrapado en el primer óptimo local. A tal efecto, dirige la búsqueda teniendo en cuenta su historia, la cual guarda en una memoria. Podría decirse que se lleva a cabo un cierto aprendizaje y que la búsqueda es inteligente. Todos los procedimientos de TS se apoyan en la afirmación de que es mejor una mala decisión basada en información que una buena decisión al azar, ya que, en un sistema que emplea memoria, una mala elección basada en una estrategia proporcionará claves útiles para continuar la búsqueda de forma eficiente, mientras que una buena elección fruto del azar no proporciona nada para posteriores acciones. Al igual que SA, TS basa su funcionamiento en ir pasando de la solución actual del algoritmo

a otra vecina localizada en su entorno, que se convierte en la actual. Sin embargo, no se considera todo el entorno de la solución, sino que TS incorpora el concepto de entorno reducido. Existen muchas formas de definir tal concepto, siendo la más sencilla la que utiliza un esquema de memoria a corto plazo, que consiste en etiquetar como tabú (prohibidas) las soluciones antes visitadas en un pasado más o menos cercano. Así, en una iteración determinada, el entorno reducido de una solución será su entorno usual menos las soluciones etiquetadas como tabú. Lo que se persigue con esto es evitar que la búsqueda entre en un ciclo. Pasadas cierta cantidad de iteraciones, se supone que la búsqueda se encontrará ya en una región distinta, por lo que puede liberarse del estado tabú a las soluciones antiguas. Para almacenar estos movimientos prohibidos se emplea lo que se conoce como lista tabú.

Un algoritmo de búsqueda tabú tiene dos parámetros básicos, que son el tiempo que un movimiento es considerado tabú y el número máximo de iteraciones que pueden pasar sin encontrar movimientos que nos lleven a una solución mejor que la actual. Si se supera dicho número, finaliza la ejecución del programa.

### *4.1.4 Algoritmos Evolutivos (EA, Evolutionary Algorithm)*

La computación evolutiva entiende la naturaleza como una máquina de resolver problemas y trata de encontrar el origen de su potencial para utilizarlo en los programas de ordenador. Podemos considerar que dicho origen se encuentra en la evolución dirigida por la selección natural de los individuos más adaptados a su entorno. Existe una diferencia conceptual entre la selección natural y la que nosotros llevamos a cabo en un algoritmo, y es que la primera no es propiamente una selección, ya que no existe ninguna inteligencia exterior que conduzca la evolución, mientras que nosotros podemos elegir a aquellos individuos que más nos interesen. Sin embargo, en ambos casos, la evolución se produce.

Realmente, el término algoritmo evolutivo hace referencia a varias técnicas, si bien hoy en día se suele utilizar este término para referirse a ellas indistintamente. Estas técnicas comprenden la programación evolutiva, las estrategias evolutivas y los algoritmos genéticos, siendo esta última la que más relevancia ha tenido. Todas ellas están inspiradas en los principios de la teoría Neo-Darwiniana [24]. Dicha teoría es una combinación de la teoría evolutiva de Charles Darwin, el seleccionismo de August Weismann y la genética de Gregor Mendel, y establece que la historia de la mayoría de la vida en nuestro planeta puede ser explicada mediante un conjunto de procesos estadísticos que actúan sobre y dentro de las poblaciones y especies [25]: la reproducción, la mutación, la competencia y la selección. La reproducción es una propiedad con la que cuentan todas las formas de vida del planeta, y de no ser así, la vida misma no tendría sentido. Por otro lado, en cualquier sistema que continuamente se reproduce a sí mismo y está en constante equilibrio, la mutación está garantizada [26]. Tanto los recursos como el espacio para albergar la vida en el planeta son finitos, lo que provoca que exista la competencia. Por último, la selección es consecuencia del exceso de organismos que han llenado el espacio de recursos disponibles. La evolución es, por lo tanto, el resultado de la interacción de estos procesos a lo largo del tiempo. Así pues, para simular el proceso evolutivo en un computador es necesario establecer una representación adecuada de los individuos que se reproducirán, las operaciones que afecten a dichos individuos, una función de aptitud que nos diga lo bueno que es cada individuo y un mecanismo de selección.

### *4.1.5 Búsqueda dispersa (SS, Scatter Search)*

Los conceptos y principios fundamentales de esta técnica fueron propuestos en la década de los setenta [27], aunque fue Laguna y Marti [28] quienes publicaron el primer libro específico acerca de ella. Según estos autores, la SS está formada por los siguientes elementos:

- Un método de diversificación que genere un conjunto de soluciones iniciales a las que se denomina agentes (en torno a 100 agentes).

- Un método para actualizar lo que se conoce como conjunto de referencia. Dicho conjunto se inicializa con una porción de las mejores soluciones del conjunto de soluciones diversas (en torno a 10 agentes). El criterio para extraer estas soluciones se basa en la calidad y la diversidad. La actualización se lleva a cabo cuando, mediante la recombinación, se encuentran agentes mejores que los existentes en el conjunto de referencia.

- Un método de generación del subconjunto, que opera en el conjunto de referencia seleccionando algunas soluciones para que sean combinadas más adelante.

- Un método para combinar las soluciones, que transforma las soluciones del subconjunto en un vector de soluciones combinadas.

- Un método de mejora basado en algún procedimiento de búsqueda local que trate de incrementar la calidad de las soluciones iniciales así como la de las obtenidas mediante la recombinación.

La búsqueda dispersa termina cuando no hay soluciones nuevas en el conjunto de referencia que se puedan combinar. Esto puede llegar a ser un inconveniente si se ha establecido un criterio de parada predeterminado, en cuyo caso se puede tratar de reconstruir el conjunto de referencia ejecutando de nuevo el procedimiento.

### *4.1.6 Procedimiento de búsqueda basado en funciones greedy aleatorizadas adaptativas (GRASP, Greedy Randomized Adaptive Search Procedure)*

GRASP [29, 30] es una meta-heurística en la cual cada iteración está formada por dos fases. La primera de ellas recibe el nombre de constructiva y sirve para crear una buena solución inicial al problema, que luego es mejorada en la segunda fase mediante un algoritmo de búsqueda local.

En cada paso de la fase de construcción se añade un nuevo objeto a la solución que se quiere producir. A la hora de elegir dicho objeto se ordenan todos los candidatos según cierta función greedy y se escoge uno de ellos al azar. Se dice que esta técnica se adapta porque los beneficios obtenidos en cada paso se actualizan al añadir el objeto seleccionado a la solución parcial. Normalmente no se suele emplear mucho tiempo en la fase de la mejora local, pues GRASP realiza muchas iteraciones para luego quedarse con la mejor solución de todas, por lo que no resulta muy beneficioso detenerse demasiado en optimizar una determinada solución.

### *4.1.7 Búsqueda en vecindario variable (VNS, Variable Neighborhood Search)*

La VNS [31] es una técnica de búsqueda local que intenta escapar de los óptimos locales permitiendo que se utilicen distintas estructuras de vecindad durante la búsqueda. Una estructura de vecindad es una función que asigna a una solución *s* un conjunto de soluciones vecinas. Visto así, cualquier procedimiento que permita pasar de una solución *s* a otra define implícitamente una estructura de vecindad, siendo *N(s)* el conjunto de todas las soluciones a las que se puede llegar desde *s* aplicando el procedimiento.

El algoritmo comienza definiendo varias estructuras de vecindad y generando a continuación una solución inicial *s*, la cual va a ser modificada haciendo uso de una de las estructuras de vecindad elegidas aleatoriamente. Se entra entonces en el bucle principal, del que saldremos cuando se cumpla cierta condición de parada, y que está formado por tres fases: agitación, búsqueda local y movimiento. En la primera se aplica la estructura de vecindad a *s*, consiguiendo una nueva solución *s'* que será la usada para la búsqueda local. De esta manera, la segunda fase produce otra solución, *s''*, que es comparada con *s*. Si es mejor, entonces *s''* reemplaza a *s* (*s = s''*, fase de movimiento) y se inicia una nueva iteración del algoritmo. Si no, se selecciona la siguiente estructura de vecindad y comienza de nuevo la fase de agitación.

Cambiar de vecindad para escapar de los óptimos locales es una idea interesante, ya que una solución que sea óptimo local en una vecindad, probablemente no lo sea en otra, ya que no todos los espacios poseen las mismas propiedades.

### *4.1.8 Optimización mediante colonias de hormigas (ACO, Ant Colony Optimization)*

ACO es una de las más novedosas meta-heurísticas y fue propuesta por Marco Dorigo [32]. Esta técnica se inspira en el comportamiento real de las colonias de hormigas. Estudios realizados explican por qué seres vivos casi ciegos, como son las hormigas, son capaces de seguir la ruta más corta entre el nido y el alimento. Esto se debe a que las hormigas pueden transmitirse información gracias a que cada una de ellas va dejando un rastro de una sustancia llamada feromona en su camino. Así, mientras una hormiga aislada se mueve de forma esencialmente aleatoria, las hormigas de la colonia detectan el rastro de feromona dejado por otras, tienden a seguirlo y lo hacen más atractivo, pues dejan a su vez su propia feromona. Sin embargo, dicho rastro se va evaporando con el tiempo con lo que la probabilidad de que sea seguido decrece.

### *4.1.9 Algoritmos meméticos (MA, Memetic Algorithms)*

La denominación memético surge del término inglés meme, acuñado por Dawkins [33] como el análogo del gen en el contexto de la evolución cultural. Por lo tanto, el concepto de MA [34] se basa en una técnica de optimización inspirada en los principios de Darwin sobre la evolución natural y la noción de meme. En la práctica, los MAs utilizan una población de individuos (denominados agentes) a los que se aplican los mismos operadores que los algoritmos evolutivos (EAs), a excepción del uso de una búsqueda local para refinar los individuos [35]. Al igual que los procedimientos basados en colonias de hormigas, los MA son técnicas muy recientes que han demostrado su gran rendimiento en problemas complejos [36]. Podemos decir a modo de resumen que son técnicas híbridas de optimización que combinan conceptos tomados de otras metaheurísticas, tales como la búsqueda en poblaciones típica de los algoritmos evolutivos o la mejora local de las técnicas de seguimiento del gradiente. En el caso de los MA, y a diferencia de los algoritmos genéticos, las soluciones reciben el nombre de agentes.

### *4.1.10 Algoritmos culturales (CA, Cultural Algorithms)*

Los algoritmos culturales [37] son técnicas de computación evolutiva que operan en dos espacios: el espacio de la población (al igual que en los algoritmos evolutivos), y el espacio de creencias, en el que se almacenan experiencias (positivas o negativas) que la población ha adquirido a lo largo del proceso de búsqueda. Esta información sirve para guiar el proceso de búsqueda y generar nuevos individuos, lo cual puede influir en cualquiera de los operadores evolutivos (cruce, mutación, etc.). El espacio de creencias representa de forma numérica intervalos para cada una de las variables de decisión, definidos por las restricciones del problema. En este caso, los intervalos reflejan la información adquirida por la población en cuanto a la región factible, que en un principio son los intervalos dados en los datos de entrada, y se actualizan con los nuevos individuos que aparecen en la población durante el proceso evolutivo.

### *4.1.11 Algoritmos de optimización basada en cúmulo de partículas (PSO, Particle Swarm Optimization)*

La optimización mediante cúmulo de partículas (Particle Swarm Optimization - PSO) es una técnica poblacional desarrollado por Kennedy y Eberhart [38], basada en un conjunto de soluciones que guían el proceso de búsqueda a través de la inteligencia colectiva de dicha población. Esta técnica de optimización está inspirada en el comportamiento social de los animales tales como las bandas de pájaros o los enjambres de abejas. PSO ha demostrado ser eficiente en problemas multidimensionales continuos.

El optimizador por enjambre de abejas (OEP), es un caso particular del PSO, nace, a semejanza de otras técnicas estocásticas de cálculo evolutivo. El objetivo de las abejas es encontrar en el campo el emplazamiento con la densidad de flores más alta. Sin ningún conocimiento a priori del campo, las abejas empiezan en sitios aleatorios con velocidades aleatorias en busca de las flores. Cada abeja puede recordar los emplazamientos donde ha encontrado más flores, y comunicándose entre ellas conoce las posiciones donde las otras abejas han encontrado un cúmulo de flores. En este punto la abeja tiene la opción de volver al lugar donde encontró el mayor cúmulo de flores o dirigirse al sitio aportado por las otras abejas donde hay una mayor densidad de flores. La abeja acaba acelerando en una combinación de ambas direcciones alterando su trayectoria para volar en dirección entre los dos puntos, dependiendo si la nostalgia o la influencia social dominan su decisión. A lo largo del camino, una abeja quizá encuentra una posición con una mayor concentración de flores delas que han sido encontradas previamente. Entonces esta posición sería señalada como la de mayor cúmulo de flores encontrada por el enjambre además del descubrimiento individual de la abeja. Las abejas constantemente comprueban el territorio que sobrevuelan hasta encontrar la mayor concentración de flores de todo el campo.

A continuación se definen los términos claves de este algoritmo:

- *Partícula*: correspondería a cada una de las abejas del enjambre. En general, todas las partículas del enjambre actúan individualmente bajo el mismo principio de gobierno: acelerar hacia la mejor posición individual encontrada y hacia la mejor posición encontrada por el conjunto mientras constantemente comprueban el valor de su posición actual.

- *Posición*: es el emplazamiento o lugar donde se encuentra la partícula dado un espacio de soluciones posibles. En el ejemplo anterior, este parámetro position correspondería al par de coordenadas x-y dentro del espacio bidimensional con el que se podría modelar el campo de flores. En general, este espacio puede ser N- dimensional, con un valor mínimo y máximo en cada dimensión.

- *Función de coste* (función Fitness): es una función que para toda posición en el espacio de soluciones devuelve un número indicativo del valor de aquella posición. Esta función es necesaria ya que el PSO se trata de una técnica computacional evolutiva y requiere de una función que evalúe la bondad de aquella posición.

- *Pbest*: en la analogía anterior cada abeja recordaba el lugar donde individualmente había encontrado el mayor número de flores. Este lugar con el fitness más alto encontrado por una abeja se conoce como personal *best* o *pbest*. Cada abeja tiene su propio *pbest* determinado por el camino por donde ha volado. En cada punto del camino la abeja compara el fitness de esa posición con su *pbest*. Si en el actual punto se encuentra un valor de fitness mayor, su antiguo *pbest* es sustituido por este nuevo.

- *Gbest*: es el punto con mayor valor de fitness, es decir, el lugar con mayor densidad de flores encontrado por el enjambre entero y, el cual, es conocido por cada abeja. Por tanto, para todo el enjambre hay un único *gbest* por el que cada abeja es atraída. En cada punto del camino recorrido por una abeja, ésta compara el fitness en el punto actual con el del *gbest*. Si en el actual punto se encuentra un valor de fitness mayor, el *gbest* es sustituido por la posición de esta abeja.

- *GBParticle*: es el bloque de información que contiene los datos de eficiencia media, fitness, posición y velocidad de la partícula que se encuentra en *gbest*.

## 4.2 Optimización multi-objetivo

Mientras que los problemas de optimización mono-objetivo requieren la optimización de una única función objetivo, los problemas de optimización multi-objetivo [39] consisten en optimizar dos o más objetivos de forma simultanea, lo cual conlleva aumentar sensiblemente la dificultad del proceso de optimización, sobre todo en los casos en los que ambos objetivos sean contrapuestos (la mejora de uno de ellos suela conllevar el deterioro del otro). Gran parte de los problemas del mundo real implican la optimización simultánea de varios objetivos que generalmente presentan conflictos entre sí, es decir, la mejora de un objetivo puede conllevar el deterioro de otro/s. Ingenieros y técnicos se enfrentan a este tipo de problemas a la hora de diseñar e implementar sistemas de todo tipo: existen múltiples objetivos a cumplir y se espera lograrlos todos en la medida de lo posible.

Mientras que en el caso mono-objetivo resulta trivial la elección de la mejor solución (aquella con valor máximo/mínimo de la función objetivo del problema de maximización/minimización a resolver), en el caso multi-objetivo la existencia de múltiples soluciones conlleva la imposibilidad de decidir automáticamente cuál de ellas es la mejor, siempre y cuando se considere a todos los objetivos con igual importancia. Existen dos alternativas a la hora de afrontar los problemas multi-objetivo. Una primera posibilidad consiste en utilizar una función matemática de suma de pesos [40] que combine los objetivos a optimizar. De esta forma, si se le da más importancia a un objetivo, bastará con darle un mayor peso al mismo dentro de dicha función matemática. Esta técnica será la que utilicemos en dicho trabajo. Otra opción sería la de aplicar optimización multi-objetivo basada en frentes de Pareto [41]. A continuación ofreceremos una visión general de ambas estrategias y algunos ejemplos de técnicas existentes:

- *Optimización multi-objetivo mediante funciones de suma de pesos*: Una de las primeras estrategias que se propusieron para lidiar con varios objetivos consiste en combinarlos mediante una función de suma de pesos [40], que se conoce habitualmente como función de agregación. Esta estrategia consiste en sumar las funciones objetivo de los diferentes objetivos a optimizar en aplicando diferentes pesos a cada objetivo a modo de ponderación. Esto significa que un problema con objetivos múltiples se transforma en un problema de optimización mono-objetivo de la forma:

$$\min \sum_{k=1}^{K} (w_k * f_k(s)); \qquad \sum_{k=1}^{K} w_k = 1 \tag{3}$$

  donde $w_k$ es el peso que representa la importancia relativa del $k$-ésimo objetivo. Puesto que los resultados de resolver un problema de optimización usando la fórmula indicada pueden variar significativamente conforme se modifiquen los pesos, y puesto que se sabe normalmente muy poco acerca de la forma más adecuada de seleccionar estos coeficientes, se suele resolver el mismo problema usando diferentes valores de $w_k$. Debe advertirse que los pesos no reflejan proporcionalmente la importancia relativa de los objetivos, sobre todo en aquellos casos en los que difieran las unidades en las que se expresen los diferentes objetivos. Esta técnica es muy eficiente desde el punto de vista de recursos de cómputo, y puede usarse para generar una solución que pueda utilizarse como un punto inicial para otras técnicas. Sin embargo, la determinación de los pesos a utilizar se convierte en un gran hándicap, sobre todo cuando no existe una información completa del problema a optimizar. Además, en muchos casos puede perder porciones cóncavas de la curva compromiso, lo cual es un inconveniente serio si se pretenden resolver ciertos problemas.

- *Optimización multi-objetivo basada en Frentes de Pareto*: Se dice que las soluciones de un problema con objetivos múltiples son óptimas porque ninguna otra solución, en todo el espacio de búsqueda es superior a ellas cuando se tienen en cuenta todos los objetivos al mismo tiempo, esto es, ningún objetivo puede mejorarse sin degradar a los demás. Al conjunto de estas soluciones óptimas se conoce

como soluciones Pareto óptimas [41]. A partir de este concepto se establece como requisito para afirmar que una situación es mejor que otra, el que ésta sea mejor en algún objetivo y no peor en otra.

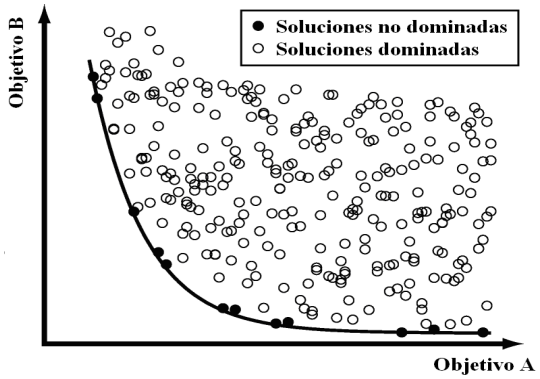


**Figura 10**. Conjunto de soluciones y frente de Pareto.

En la Figura 10 se observa un conjunto formado por un ámplio número de soluciones de un problema bi-objetivo, donde sólo aquellas sombreadas son soluciones no dominadas y próximas al frente Pareto-óptimo, mientras que la mayoría de las soluciones son dominadas, es decir, para toda solución no sombreada (dominada), existe al menos una solución sombreada (no dominada) tal que es mejor en un objetivo sin empeorar en el otro. Obviamente, el objetivo de cualquier algoritmo multi-objetivo basado en Pareto-dominancia radica en encontrar el mayor número posible de soluciones que se encuentren en el frente óptimo de Pareto. A continuación se definen los conceptos de dominio y optimización de Pareto, aplicados a un problema de minimización; la extensión al caso de un problema de maximización es trivial.

**Dominancia de Pareto:** En el caso de un problema de minimización, y siendo *k* el número de objetivos, se dice que un vector (solución) $u=(u_1,...,u_k)$ domina a otro $v=(v_1,...,v_k)$, ($u \prec v$) si y sólo si:

$$\forall i \in [1,..,k] : u_i \leq v_i, \exists i \in [1,..,k] : u_i < v_i \qquad (4)$$

Dos vectores (soluciones), $u=(u_1,...,u_k)$, $v=(v_1,...,v_k)$, son indiferentes ($u \sim v$) si ninguna de ambas domina a la otra.

**Optimalidad de Pareto:** Sea *Z* el espacio de búsqueda, se dice que una solución $u=(u_1,...,u_k)$ es Pareto-óptima si y sólo si no existe otra solución, $v \in Z$, tal que *v* domine a *u* ($v \prec u$). Esto significa que el punto *u* es un óptimo de Pareto si no existe un vector *v* que mejore a *u* en alguno de los objetivos sin empeorar de forma simultánea en los otros.

En general, la solución obtenida para el problema de optimización multiobjetivo no será única: estará formada por el conjunto de todos los vectores no dominados, a los que se conoce con el nombre de conjunto de no dominados o frente de Pareto. Por tanto, una métrica intuitiva para determinar la calidad de las soluciones obtenidas sería la proximidad de dichas soluciones a las soluciones Pareto-óptimas. Por desgracia, para largas instancias de problemas de optimización multi-objetivo, especialmente aquellos incluidos en la categoría de problemas NP-duros, resulta imposible conocer las verdaderas soluciones del frente de Pareto, razón por la cual el objetivo será obtener las mejores soluciones en términos de minimización de los objetivos a tratar.

A continuación vamos a comentar brevemente algunas de las heurísticas multi-objetivo más conocidas [19]. La primera aproximación multi-objetivo basada en optimización de Pareto fue propuesta por David Goldberg en 1989 [41]. Goldberg presentó MOGA, siendo éste el primer algoritmo que usó asignación de aptitud basada en el óptimo de Pareto, la cual se propuso con la idea de resolver los problemas de la técnica de Schaffer. Goldberg sugirió el uso de asignación de jerarquías y selección basadas en no dominancia para trasladar la población hacia el frente Pareto-óptimo.

Fonseca y Fleming [42] propusieron una técnica en la cual la jerarquía de un cierto individuo corresponde al número de cromosomas por los cuales es dominado en la población actual. Con posterioridad a MOGA, Srinivas y Deb propusieron el algoritmo NSGA [43]. Su funcionamiento se basa en el uso de varias capas de clasificación de los individuos. Posteriormente, se desarrolló el algoritmo elitista multi-objetivo NSGA-II [44] mejorando a su antecesor. Casi simultáneamente a NSGA, Horn junto con otros investigadores propusieron el NPGA [45]. Este algoritmo lleva a cabo un proceso de selección mediante torneo basado en dominancia de Pareto.

Posteriormente Zitzler y Thiele presentaron Strength Pareto Evolutionary Algorithm (SPEA). SPEA [46] es considerado como un algoritmo multi-objetivo de segunda generación. Los MOEAs de segunda generación se caracterizan por la introducción del concepto de elitismo. Poco tiempo después de la presentación de SPEA, y con el objetivo de superar las limitaciones que algoritmos posteriores pusieron de manifiesto, Zitzler junto con otros investigadores presentaron SPEA2 [47].

Corne y Knowles propusieron Pareto-Envelope based Selection Algorithm (PESA) [48]. Se trata de un algoritmo híbrido entre PAES y SPEA. Pero tras analizar algunas posibles mejoras sobre PESA, los mismos autores propusieron PESA II [49].

También se han propuesto otras alternativas basadas en algoritmos genéticos multi-objetivo, como los micro-GA [50], consistentes en utilizar una población de tamaño reducido (cinco individuos), una elevada probabilidad de cruce (100%) y una probabilidad de mutación nula. Con posterioridad, Toscano y Coello propusieron el llamado micro-GA2 [51].

Recientemente se han propuesto nuevos procedimientos, como MOSATS (Multi-objective Simulated Annealing and Tabu Search) [52], que es una técnica poblacional basada en combinar enfriamiento simulado y búsqueda tabú. En concreto, cuando la temperatura de enfriamiento va descendiendo, también lo hace la probabilidad de aceptar soluciones dominadas por otras encontradas previamente, a la vez que se hace uso de una lista de movimientos tabú para evitar que la búsqueda sea cíclica. Otra meta-heurística multi-objetivo es MOSSSA (Multi-objective Scatter Search with Simulated Annealing) [53] que hace uso de una población a la que se le aplica búsqueda dispersa, y que durante el proceso de búsqueda local (incluido en la búsqueda dispersa) se utiliza enfriamiento simulado en lugar de un optimizador local basado en descenso de colinas. Otros autores han aplicado también recientemente la búsqueda dispersa en términos multi-objetivo [54]. Recientemente, destacar los trabajos de Talbi y Coello [55] en los que proponen nuevas estrategias evolutivas híbridas para resolver problemas

de optimización multi-objetivo. Por último destacar el trabajo de Schütze y Coello [56] en el que desarrollan un nuevo algoritmo memético para la optimización multi-objetivo.

# 5. TP-MOEA: ALGORITMO EVOLUTIVO APLICADO AL PROBLEMA DE EMPAQUETAMIENTO DE PALETS

Los problemas reales usualmente requieren la búsqueda de soluciones que satisfagan de forma simultánea múltiples criterios de desempeño u objetivos, los cuales pueden ser contradictorios. Cuando es factible combinar los objetivos de un problema de manera adecuada, es posible considerar un único objetivo a optimizar. En este caso, para obtener la solución del problema basta con encontrar el mínimo o el máximo de una única función que resume todos los objetivos que se desean optimizar. Sin embargo, lo usual es que no se conozca la manera óptima de combinar los diferentes objetivos o esta sea inadecuada, cuando no imposible hacerlo. En este caso, se dice que el problema es un Problema de Optimización Multiobjetivo (Multiobjective Optimization Problem - MOP) [44,46]. En problemas de optimización multiobjetivo con objetivos contradictorios no siempre existe una solución única que pueda ser considerada como la mejor, sino un conjunto de soluciones que representan los mejores compromisos entre los distintos criterios. Dicho conjunto es llamado conjunto Pareto-óptimo y su imagen en el espacio objetivo, es denominado frente de Pareto [41]. Los Algoritmos Evolutivos (Evolutionary Algorithms - EAs) han demostrado ser especialmente adecuados para la optimización multiobjetivo. La literatura actual reporta un gran número de Algoritmos Evolutivos para Optimización Multiobjetivo (Multiobjective Evolutionary Algorithms - MOEA). El algoritmo evolutivo multi-objetivo implementado para el problema de empaquetamiento de palets en camiones se denomina TP-MOEA. Con la aplicación cada vez más extendida de MOEAs en problemas reales de optimización, se hace necesario mejorar el desempeño de los mismos a fin de asegurar la aplicabilidad de la técnica en problemas de complejidad creciente. Para ello, una alternativa es la incorporación de conceptos de paralelismo al diseño de estos algoritmos. Si bien las ideas de paralelización de EAs han estado presentes desde prácticamente sus orígenes y existen varios modelos de paralelización para algoritmos evolutivos monobjetivo, estos modelos apenas han sido integrados al campo de la optimización evolutiva multiobjetivo. A fin de lograr una mejora en la capacidad de búsqueda de los MOEAs, el presente trabajo propone un marco para el desarrollo y aplicación de algoritmos evolutivos multiobjetivo paralelos (parallel Multiobjective Evolutionary Algorithms - pMOEAs). Siguiendo el marco propuesto, se han desarrollado distintos pMOEAs y se ha aplicado a funciones de prueba de dificultad creciente.

## 5.1 Paralelización del algoritmo evolutivo

Para la mayoría de los problemas, el conocimiento del Frente Pareto óptimo ayuda al tomador de decisiones a seleccionar aquella solución que representa el mejor compromiso. Generar dicho frente puede ser computacionalmente costoso o incluso imposible, en especial en problemas reales de ingeniería. Entonces, lo único que se puede pretender es obtener una buena aproximación al frente Pareto optimo verdadero. Los algoritmos evolutivos multiobjetivo son una alternativa practica en la búsqueda de soluciones de compromiso para problemas reales donde los métodos exactos son inaplicables o ineficientes. Durante la ejecución de un algoritmo evolutivo basado en dominancia Pareto, en cada generación se debe encontrar un conjunto de soluciones Pareto óptimas con respecto a la población genética actual es encontrado en cada generación. La integración de la computación paralela y la computación evolutiva da origen a los algoritmos evolutivos paralelos multiobjetivo (pMOEAs). La utilización de pMOEAs posee varias ventajas con respecto a otros métodos uniendo las características propias de los MOEAs con las ventajas del cómputo paralelo. Básicamente, los pMOEAs intentan encontrar soluciones tan buenas o mejores que sus contrapartes secuenciales en menos tiempo y/o explorar un espacio mayor de posibles soluciones. Básicamente, los modelos de paralelización de EAs más importantes son: el modelo maestro-esclavo, el de difusión y el de islas. Cantu-Paz [57] presenta una revisión de estos modelos aplicados a optimización monobjetivo, mientras que Veldhuizen [58] lo hace en un contexto multiobjetivo. El esquema de paralelización utilizado en el presente trabajo para el desarrollo de MOEAs paralelos se basa en el modelo de islas. El modelo de islas está basado en el fenómeno natural de que las poblaciones se encuentran, usualmente, relativamente aisladas unas de otras [59]. En el modelo de islas una población se divide en subpoblaciones separadas e independientes, llamadas islas. Los diferentes operadores evolutivos trabajan en cada isla, lo que implica que las distintas poblaciones se encuentran explorando en regiones diferentes del espacio de búsqueda. Cada isla también podría tener distintos parámetros así como diferente estructura de MOEA. Además, individuos de una isla podrían migrar a otra isla de acuerdo a algún criterio. El modelo de islas requiere la selección de una política de migración que señale: la manera en que los individuos migrarían, el número de migrantes, la frecuencia de migración, de dónde se seleccionarán los elementos a migrar y cómo se realizará el reemplazo de los elementos en una población por los migrantes provenientes de otras poblaciones. Además, es preciso definir los distintos parámetros y algoritmos a utilizar en cada una de las diferentes islas. Definiendo convenientemente la política de migración, el modelo de islas puede aplicarse a varias arquitecturas paralelas, especialmente en sistemas paralelos de memoria distribuida. Primeramente, se establece un criterio para la selección de soluciones a enviar. En este trabajo se ha decidido seleccionar como migrantes las soluciones no dominadas de la población en el momento que la migración ocurre. Un parámetro limita el número máximo de soluciones a enviar. Además, se define un criterio que determinará la frecuencia en que ocurrirá la migración. Se han explorado varias opciones como: enviar individuos transcurrido un número de generaciones, enviar individuos de forma probabilística, enviar individuos de forma adaptativa, entre otras. En el primer caso, se determina el número de generaciones que deben transcurrir para que se produzca un envío. En el segundo, se proporciona como parámetro del algoritmo la probabilidad de que un envío ocurra en una determinada generación. La tercera opción, se basa en modificar la frecuencia de migración conforme el proceso de evolución ocurre. En el presente trabajo, tanto el envío como la

recepción de las soluciones se manejan en forma asíncrona. Esto permite que los PMOEAs puedan continuar inmediatamente después que han ejecutado una primitiva de envío así como la verificación, sin espera, de la recepción de soluciones. Además de la reducción en el tiempo de espera, la comunicación asíncrona es adecuada para lidiar con la posible pérdida de datos y elementos de procesamiento. En cuanto a la forma en que procederá el reemplazo de elementos de la población por migrantes, habiendo explorado varias opciones, se ha optado por el reemplazo de las soluciones con peor objetivo 1 por los elementos recibidos.

La topología del vecino más cercano se propuso para extender las mejores soluciones entre las diferentes islas como muestra la figura 11.

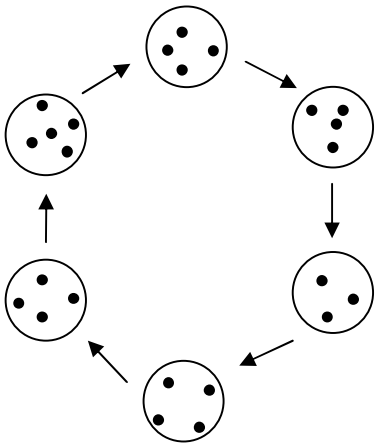

**Figura 11.** Topología de migración del vecino más cercano.

El algoritmo de paralelización multi-objetivo implementado denominado PTP-MOEA está inicializado con cuatro islas. Cada isla es inicializada de forma aleatoria insertando los palets de cada cliente en un camión. Al añadir un nuevo objetivo al problema se han diseñado distintos operadores que mejoran el balanceo de carga de los contenedores. El método de migración se lleva a cabo cada 200 iteraciones, donde todas las islas envían las soluciones no dominadas del frente de Pareto a sus vecinos, los cuales reemplazan sus soluciones dominadas por estas.

Además los operadores de recombinación y mutación utilizan el mismo optimizador en cada isla con el objetivo de reducir el número de contenedores insertando objetos en los espacios disponibles de los contenedores menos ocupados.

Finalmente cuando se alcanza la condición de parada, la ejecución de la isla se para y se genera un nuevo frente de Pareto con todas las soluciones no dominadas de cada isla. Las peores soluciones para el objetivo 1 son reemplazas manteniendo una población constante.

Esta implementación paralela se ha llevado a cabo mediante múltiples hilos en Java, en un procesador Intel Core i7-720QM (1,6GH) con ocho hilos.

## 5.2 Estructura del algoritmo evolutivo para el problema de empaquetamiento

La presente investigación propone un algoritmo evolutivo para resolver el problema de empaquetamiento anteriormente descrito. El algoritmo consiste en una población de individuos los cuales son optimizados usando un conjunto de operadores típicos evolutivos (la mutación, el cruce, y la selección) y un optimizador. Estos algoritmos toman algunas ideas propuestas por otros autores, como el empleo de una lista de espacios máximos disponibles en los contenedores, para mejorar el funcionamiento de los operadores. En los próximos apartados detallaremos la estructura y el funcionamiento del algoritmo.

### *5.2.1 Ejemplo y estructura de datos para 2DPP*

Para comprender el problema de empaquetado, la figura 12 muestra un ejemplo de representación del problema. En este caso intentamos almacenar un conjunto de 10 piezas en el menor número posible de contenedores. En el caso del empaquetamiento de palets se hace relativamente más sencillo al tener todas las piezas el mismo tamaño.

Como podemos observar en la Figura 12, para empaquetar todas las piezas se han necesitado dos contenedores y se han generado un total de tres espacios. Para almacenar toda esta información necesitaremos las siguientes estructuras de datos:

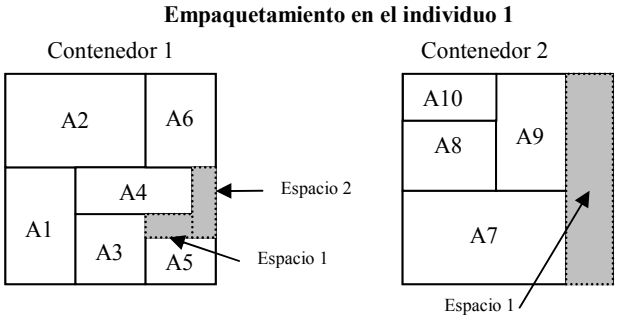


**Figura 12.** Ejemplo empaquetamiento 2D.

*Objeto pieza:* este objeto estará formado por tres variables fijas como son el ancho, largo y peso de la pieza que no varia, y tres variables que iremos modificando conforme movamos una pieza de sitio. Estas son las posiciones $x$ e $y$ de la pieza en el contenedor y el número de este en la que se encuentra la pieza.

*Vector de objetos:* En esta estructura almacenaremos para cada individuo un vector de objetos (piezas) para saber donde están todas las piezas insertadas.

Por otro lado necesitamos una estructura para almacenar todos los espacios disponibles en los diferentes contenedores. Para ello hemos implementado una serie de vectores anidados como se explica a continuación.

*Espacio:* el espacio será un objeto que contiene cuatro variables necesarias para identificarlo en un contenedor. Estas son las posiciones $x$ e $y$ en el contenedor, así como el ancho y largo del espacio.

*Vector de espacios de un contenedor:* almacenaremos para cada contenedor una lista con sus espacios disponibles. Por lo tanto podremos tener para un solo contenedor una lista de espacios disponibles.

*Vector de espacios para cada individuo:* estará formado por una matriz donde se almacenara, para cada individuo y para cada bin de ese individuo los espacios que tiene disponibles.

## 5.3 Funcionamiento general del algoritmo evolutivo

A continuación vamos a sintetizar la implementación de un algoritmo evolutivo. Además de describir el pseudocódigo, en esta sección se persigue que el lector comprenda mediante explicaciones sencillas, pero detalladas, cómo trabaja un algoritmo evolutivo. La función tiene como entrada un conjunto de parámetros de configuración, como pueden ser el tamaño de la población, la cantidad de padres o el número de hijos a crear. De todas formas, esto no debe importarnos ahora, pues no influye para el buen entendimiento del algoritmo y sólo tiene interés a la hora de una implementación real. La salida, como queda indicado mediante la sentencia Return, puede ser la mejor solución encontrada o un frente de soluciones.

Si observamos la Figura 13, veremos que un algoritmo evolutivo mantiene en todo momento una población de individuos (soluciones al problema que se trata de resolver) que se relacionan entre sí (reproducción) en un marco competitivo (selección y actualización). Pasemos a explicar por separado cada una de las partes que componen el algoritmo anterior.

**Algoritmo Evolutivo**

**Variables locales**

**Pob, Hijos, Padres: arrays de individuos**

```
Pob ← IniciarPoblación(TamañoPob);
While NOT TerminaciónEA()
    Pob ← RealizarMutación(Pob)
    Padres ← SeleccionarNmejores(Pob);
    Hijos ← Reproducir(Padres);
    ActualizarPoblación(Pob, Hijos);
    If Convergencia(Pob) Then
        ReiniciarPoblación(Pob);
Return mejor solución o frente
```

**Figura 13.** Funcionamiento general de un evolutivo.

### *5.3.1 Inicio de la población*

La primera de las funciones requeridas por el procedimiento principal del algoritmo evolutivo es la referente al inicio de la población, la cual pasamos a describir:

**Iniciar la Población**

**Variables locales**
Individuo: individuo
Pob: array de individuos

```
For j ← 1 To TamañoPob
    Individuo ← Individuo Aleatorio();
    Optimizador(Individuo);
    Pob(j) ← Individuo;
Return Pob
```

**Figura 14.** Función orientativa para el inicio de la población.

Esta función se encarga de crear la población inicial de individuos que servirá como punto de partida al algoritmo evolutivo. Dichos individuos pueden ser aleatorios, como es nuestro caso. Por otra parte, es una buena idea el emplear en cada nuevo individuo generado un optimizador que explore mejoras en él. Este optimizador no es más que un operador de mutación "dirigido" y es uno de los rasgos más distintivos de los evolutivos. Su funcionamiento sería el siguiente: realiza un cambio en el individuo. Si el individuo ha mejorado, la modificación se guarda, si no, ésta se desecha. Podemos aplicarlo un número de iteraciones fijo, un cierto número de iteraciones sin mejora, hasta alcanzar una mejora suficiente, etc. La Figura 15 muestra el funcionamiento de este operador. Cabe destacar que la función objetivo *F* corresponde a la función guía, también llamada función de evaluación, de mérito o de aptitud, que se encarga de cuantificar cuán bueno es el individuo pasado como parámetro. La inclusión de una función de este tipo es obligada en cualquier algoritmo de optimización, pues se podría decir que es el motor que los mueve. Por supuesto, la función no siempre será la misma, sino que dependerá del problema que se quiera resolver, y de su buen diseño dependerán en gran medida los resultados que se obtengan.

**Procedimiento Optimizador (Individuo)**

**Variables locales**
nuevo: individuo

```
Repeat
    nuevo ← AplicarOperador(Individuo);
    If F(nuevo) es mejor que F(Individuo) Then
        Individuo ← nuevo;
Until TerminaciónOptimizadorLocal();
Return Individuo
```

**Figura 15.** Algoritmo optimizador genérico.

La forma de operar que tiene el optimizador es similar a la de los algoritmos de ascenso de colinas. Como ya se comentó anteriormente, un algoritmo de este tipo comienza con una solución al problema, generalmente elegida al azar, y realiza una mutación de la misma. Si se produce una mejora, se conservan los cambios y se repite el proceso. En caso contrario, se descartan y el algoritmo acaba. Si bien el funcionamiento no es exactamente el mismo, la idea subyacente sí que lo es.

### *5.3.2 Selección de los mejores individuos*

La función *SeleccionarNmejores* se encarga de tomar una muestra de los *N* mejores individuos de la población, que serán los que intervengan en el proceso de reproducción. A la hora de decidir cuándo un individuo es mejor que otro se utiliza, como

ya dijimos antes, lo que se conoce como función guía. La selección es, junto con la actualización, la responsable de forzar la competición entre individuos. La fase de selección de individuos se ha implementado de acuerdo a una técnica de selección que nos ayuda a mantener la diversidad de la población buscando una convergencia hacia las mejores soluciones. El procedimiento consiste en seleccionar las soluciones no dominadas del frente de Pareto y pasarlas a las sucesivas generaciones sustituyendo a las peores soluciones de acuerdo a un criterio establecido. En este trabajo se consideran peores soluciones a aquellas soluciones dominadas con mayor número de contenedores utilizados para almacenar los palets, esto se debe a que intentaremos quedarnos con las soluciones que tengan menor número de contenedores y mejor balanceo, descartaremos aquellas que tienen un elevado número de contenedores.

### *5.3.3 Reproducción*

Es en la fase de reproducción donde se llevan a cabo los procesos de cooperación entre individuos (usualmente dos, aunque nada impide que sean más) al construir nuevos empleando información extraída del grupo de individuos elegidos para tal fin. Esto es lo que se conoce como recombinación. A cada individuo se le pueden aplicar justo después de su nacimiento operadores de mutación y un optimizador como el que comentamos anteriormente. La Figura 16 muestra gráficamente el proceso de creación de un hijo:

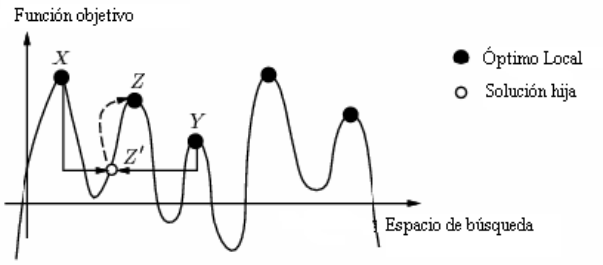


**Figura 16**. Representación grafica de un hijo.

Como se puede observar, los ejes representan el espacio de búsqueda a explorar. Los puntos que aparecen representan soluciones actuales y cuanto más altos se encuentren, mejor será su fitness o aptitud (dicho de otro modo, la solución es mejor). Los puntos en negrita son óptimos locales. Esto quiere decir que es la mejor solución dentro de su entorno, pero no la mejor solución al problema, que sería lo que se conoce como óptimo global. El punto blanco *Z′* es un individuo hijo nacido a partir de la recombinación de *X* e *Y*, que se transforma en *Z* después de la aplicación de un optimizador.

Esta situación que hemos observado aquí será la que usualmente se nos presente. Ocurre que tras la recombinación de algunos individuos para la creación de uno nuevo, éste aparecerá fuera del subespacio de los óptimos locales y se usará un optimizador que "reparará" a ese individuo hijo para intentar convertirlo en un óptimo local. Por tanto, podemos decir que los algoritmos evolutivos buscan en el subespacio de los óptimos locales con la esperanza de llegar a encontrar, si no el óptimo global, sí una muy buena solución. Para terminar este apartado, la Figura 17 muestra un algoritmo para la reproducción con la intención de eliminar cualquier duda que todavía pueda quedar, mostrando el funcionamiento clásico de este operador.

| **Función Reproducir** |
|---|
| **Variables locales**<br>padre, madre, hijo: individuos |
| **For** i ← 1 **To** NúmeroHijos<br>  padre ← ElegirPadre(población);<br>  madre ← ElegirPadre(población);<br>  hijo ← Recombinar(padre, madre);<br>  OptimizadorLocal(hijo);<br>  Hijos(i) ← hijo;<br>**Return** Hijos |

**Figura 17.** Algoritmo general para la reproducción.

### *5.3.4 Actualización de la población*

La actualización, también conocida como reemplazo, incide en el aspecto competitivo al encargarse de la importante tarea de limitar el tamaño de la población, es decir, eliminar algunos individuos para permitir la entrada de otros nuevos. En este proceso también se emplea la información proporcionada por la función guía para decidir de qué individuos se prescindirá, sustituyendo los peores individuos de la población por los mejores hijos.

### *5.3.5 Convergencia del algoritmo y reinicio poblacional*

Otro componente básico de los algoritmos evolutivos es el reinicio de la población, ya que de él dependerá que se haga un buen uso de los recursos computacionales o se desperdicien al seguir explorando soluciones en una población degenerada, o sea, con una gran similitud entre todos los individuos. Esto se conoce como convergencia del algoritmo evolutivo, y puede ser cuantificado utilizando medidas de Teoría de la Información, como por ejemplo la entropía de Shannon [60]. Cuando la entropía cae por debajo de cierto umbral, los individuos de la población se sustituyen por otros, existiendo la posibilidad de conservar una cantidad de ellos en la nueva población.

Una posible implementación podría ser la de la Figura 18. Tal y como se muestra, primero se guardan algunos individuos de la población actual en la nueva, para posteriormente completar los huecos con otros individuos creados de manera aleatoria.

Por otro lado, decir que el mecanismo de reinicio también nos puede servir para escapar de los óptimos locales. Normalmente, cuando la población llega a un estado degenerado es porque se ha alcanzado un óptimo local y el algoritmo no es capaz de salir de él, creándose entonces más y más copias del mejor individuo. Lo más adecuado en ese momento es limpiar de duplicados la población y empezar de nuevo, conservando algunas de las mejores soluciones.

**Procedimiento ReiniciarPoblación**

```
For i ← 1 To individuos a conservar
    NuevaPob(i) ← SiguienteMejor(Pob)
For i ← individuos a conservar + 1 To TamañoPob
    NuevaPob(i) ← IndividuoAleatorio()
    Optimizador(NuevaPob(i))
Pob ← NuevaPob
```

**Figura 18.** Algoritmo orientativo para el reinicio.

### *5.3.6 Condición general de parada*

Uno de los aspectos a determinar en todo algoritmo es la condición de parada. En el caso del evolutivo esto viene representado por *TerminaciónEA*, existiendo varios criterios al respecto. Por ejemplo, podemos ejecutar el algoritmo una cantidad fija de iteraciones, a las que llamaríamos generaciones, u obligar a que pase un determinado número de veces por la función guía. Esta segunda opción es más usada por la mayoría de los autores al ser más justa para comparar distintos algoritmos. Para nuestro problema estableceremos la condición de parada a un valor de 25,000 evaluaciones.

### *5.3.7 Obtención de resultados*

Para obtener el mejor individuo de todas las generaciones por las que ha ido recorriendo nuestro algoritmo, se van almacenando en una estructura el individuo con menor número de contenedores alcanzado. De esta forma nos garantizamos que ninguna solución pueda perderse, tras aplicar cualquier operador. Finalmente, almacenamos en un archivo de salida toda la población de individuos final, con la configuración de cada uno. Guardando en este archivo las posiciones de las piezas en los diferentes contenedores.

## 5.4 Descripción de los operadores utilizados

La figura 19 muestra el organigrama de nuestro algoritmo evolutivo TP-MOEA, que empieza inicializando la población de individuos, cada uno de los cuales es generado considerado una solución del problema, p. ej. un conjunto de contenedores donde todas las piezas son almacenadas. Esta inicialización consiste en incluir los palets de cada cliente en un camión hasta que no haya espacio para albergar todos los palets del cliente, en este caso, se insertan los palets en un nuevo camión. De esta forma los palets de un cliente no podrán estar repartidos en diferentes camiones cumpliendo unas de las restricciones del problema. Mientras la condición de parada no se cumpla, se le aplica a los individuos los operadores de mutación, cruce y búsqueda con una probabilidad dada. Cada vez que aplicamos a un individuo un operador, la distribución de las piezas en los contenedores varia y la lista de espacios disponibles se actualiza. Cuando la condición de parada se cumple, el algoritmo evolutivo termina y devuelve los mejores individuos.

Como se muestra en la Figura 19, el algoritmo evolutivo aquí presentado (TP-MOEA) aplica mutación, cruce, selección y un optimizador. Se ha optado por la implementación de tres operadores de mutación diferentes, lo cual añade un mejor funcionamiento del algoritmo al trabajar estos de forma conjunta, consiguiendo una gran diversidad de individuos en la población. Como puede verse en la Figura 19, los operadores van ejecutándose dependiendo de la evaluación de la función de probabilidad. Por otro lado, una función nos irá calculando los nuevos espacios máximos generados, tras aplicar cualquier operador. Finalmente, tras cumplir la condición de terminación, obtendremos en un archivo las mejores soluciones obtenidas.

A continuación explicamos el funcionamiento de cada uno de los operadores evolutivos implementados, así como el optimizador.

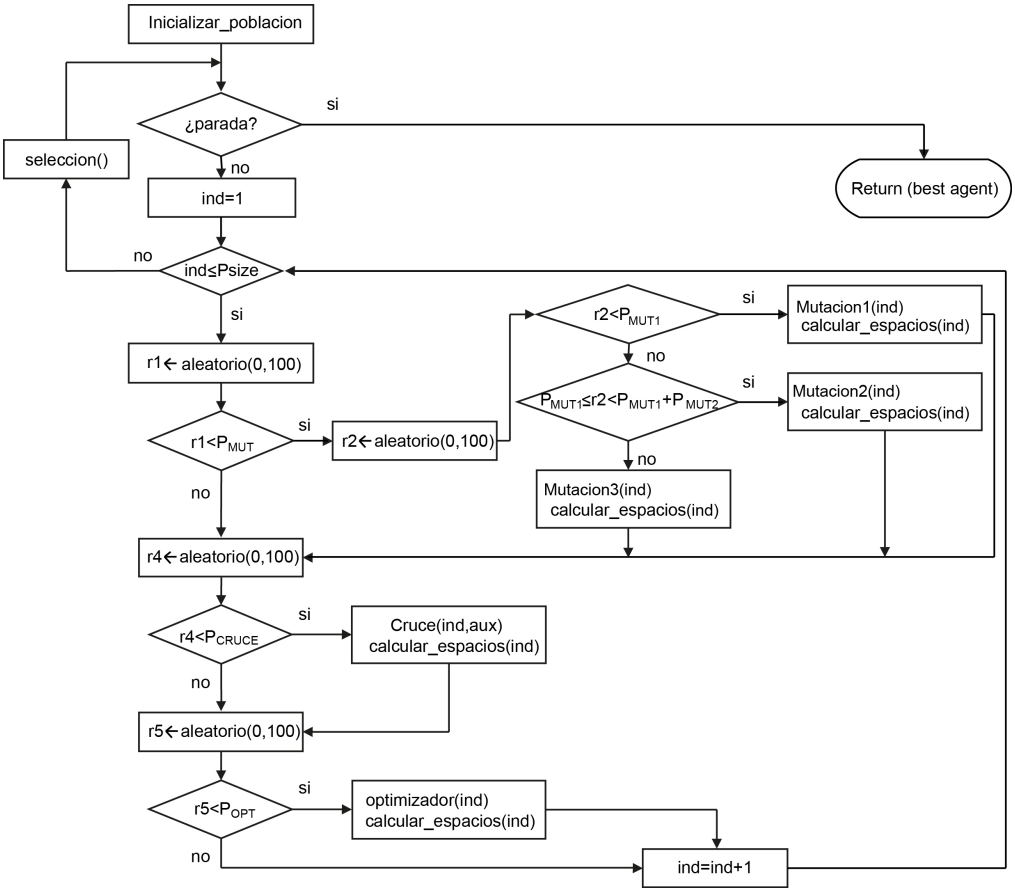


**Figura 19.** Estructura del algoritmo evolutivo para el caso bi-dimensional.

### *5.4.1 Operador de Mutación 1*

Selecciona de forma aleatoria dos clientes que estén en distintos contenedores, e intercambia todos los palets de los clientes, siempre que haya espacio en ambos contenedores. De esta forma se puede variar en un individuo el balanceo de carga de ambos contenedores.

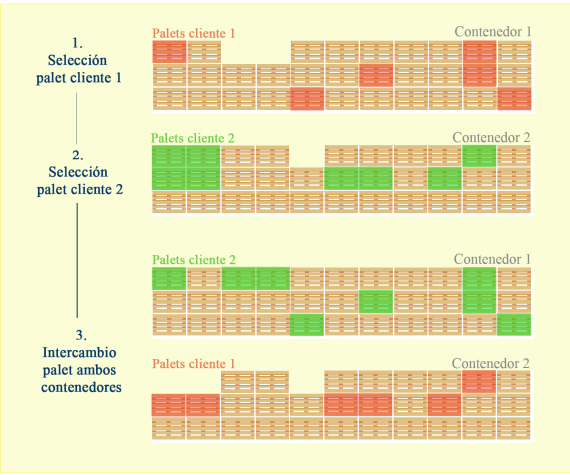


**Figura 20.** Funcionamiento operador de mutación 1.

La Figura 20, muestra un ejemplo de uso del operador de mutación 1 en la cual se seleccionan primero los palets del cliente 1 representados en rojo, posteriormente se seleccionan los palets del cliente 2 en verde. En este momento se calcula el espacio resultante de intercambiar los palets entre los distintos contenedores y si hay espacio suficiente para realizar el cambio se lleva a cabo.

### *5.4.2 Operador de Mutación 2*

Este operador selecciona de todos los contenedores uno de forma aleatoria, e intercambia dos palets de posición. De esta forma al mover dos palets de sitio se varía el balanceo de carga del contenedor.

### *5.4.3 Operador de Mutación 3*

Se selecciona un cliente de forma aleatoria, y se intenta mover sus palets a otro contenedor que tenga espacio suficiente para albergar todos los palets. Este operador puede conseguir reducir el número de contenedores en el caso de que el contenedor de donde nos llevamos los palets se quede vacío. Además varía el centro de carga de los contenedores, que se ven afectados por el movimiento de palets.

### *5.4.4 Operador de Cruce*

Dos individuos (A1, A2) son seleccionados como padres. El individuo hijo (CH) está formado por la combinación de los contenedores de los dos padres. Inicialmente, CH coge el contenedor más lleno de A1, más el resto de los contenedores de A2, pero

descartando los palets que ya están en el contenedor de A1 para no duplicar estos palets.

### *5.4.5 Optimizador*

Busca el contenedor que este más vacío, selecciona de forma aleatoria un cliente de este contenedor, e intenta insertar todos sus palet en otro contenedor para intentar reducir el número total de contenedores. El optimizador es el operador encargado de intentar mejorar el objetivo 1, intentando reducir constantemente el número de contenedores.

## 5.5 Almacenamiento dinámico de los espacios disponibles en los contenedores

La dificultad principal de aplicar los operadores para reducir el número de contenedores en el 2DPP, es el de determinar las soluciones factibles e irrealizables, p. ej. si realmente una pieza puede ser insertada en un contenedor sin que haya solapamiento con las demás piezas ya insertadas. Este problema se complica aun más cuando los contenedores están prácticamente llenos, en este caso todos los intentos de insertar la pieza podrían ser irrealizables, y así desperdiciar una gran cantidad de tiempo. Con el objetivo de resolver este problema, y por lo tanto acelerar el uso de los operadores, se ha implementado una estructura que almacena de forma dinámica una lista con los espacios disponibles (libres) rectangulares de cada contenedor. Esta estrategia es especialmente útil cuando intentamos insertar una pieza en un contenedor que esta casi lleno, siendo las áreas disponibles muy pocas y pudiendo comprobar rápidamente si se puede insertar la pieza en este contenedor. Cada vez que aplicamos un operador evolutivo, debemos actualizar la lista de espacios disponibles. Los espacios libres calculados, son el resultado de explorar cada contenedor buscando los espacios máximos disponibles, pudiendo estar estos solapados entre sí. La figura 16 muestra un ejemplo de la distribución de las piezas en un contenedor así como los espacios máximos disponibles.

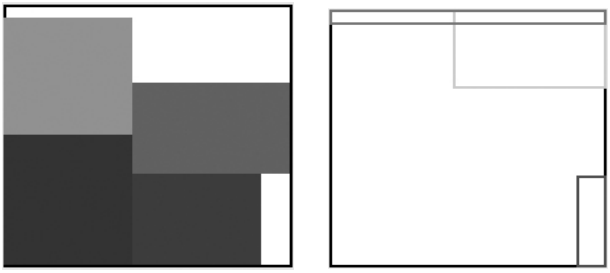

**Figura 21.** Almacenamiento de las piezas en un contenedor (izquierda), y espacios máximos disponibles (derecha).

# 6. RESULTADOS OBTENIDOS

En esta sección vamos a analizar los datos obtenidos tras ejecutar nuestro algoritmo evolutivo TP-MOEA en un conjunto de instancias de distintos tamaños para el empaquetamiento de palets en camiones. Los resultados obtenidos mostraran el buen funcionamiento del algoritmo para resolver este problema de empaquetamiento mostrando distintas soluciones para resolver el problema de la forma más eficiente posible.

## 6.1 Configuración de los parámetros

El buen funcionamiento del algoritmo depende en gran medida de la selección de los parámetros adecuados para los distintos operadores del algoritmo evolutivo. Para ello hemos realizado una batería de pruebas que nos han ayudado a afinar el porcentaje de uso de los operadores. Inicialmente probamos la eficacia de los tres operadores de mutación ejecutándolos de forma independiente para luego comparar cuál funcionaba mejor. Una vez realizada esta prueba ejecutamos de nuevo el algoritmo con los tres operadores de mutación trabajando con la misma probabilidad, y modificando en un 5% la probabilidad del operador que conseguía mejorar alguna solución. De esta forma pudimos corroborar que el segundo operador conseguía mejorar las soluciones en mayor número de ocasiones que los otros dos. De esta forma al tener tres operadores de mutación, existe una probabilidad de ejecutar mutación de un 10% y en el caso de que se aplique mutación la probabilidad de que se ejecute uno de los tres operadores es de: ($P_{mutation1}$=25%, $P_{mutation2}$=25%, $P_{mutation3}$=50%). Posteriormente, con las probabilidades de mutación fijadas fuimos alternando la probabilidad de cruce hasta llegar a un valor comprometido del 50% de probabilidad de ejecutar este operador. Por último, el algoritmo fue ejecutado para ver cómo afectaba la probabilidad del optimizador, observando como con altas frecuencias de utilización de este operador el algoritmo llegaba antes a soluciones óptimas. La tabla 1 muestra las ejecuciones realizadas con los distintos operadores de mutación.

| Número de Ejecución | *Prob.(%) OpMutacion1* | *Prob.(%) OpMutacion2* | *Prob.(%) OpMutacion3* |
|---|---|---|---|
| 1 | 25 | 25 | 50 |
| 2 | 100 | 0 | 0 |
| 3 | 0 | 100 | 0 |
| 4 | 0 | 0 | 100 |

**Tabla 1.** Parámetros para distintas ejecuciones

Un total de 25 ejecuciones independientes han sido lanzadas para cada instancia para obtener datos lo más fiables posibles a la realidad. La figura 22 muestra los frentes de Pareto de varias ejecuciones, como puede observarse la primera ejecución muestra mejores resultados que las anteriores al combinar los tres operadores, de esta forma se consigue explorar un mayor espacio de búsqueda.

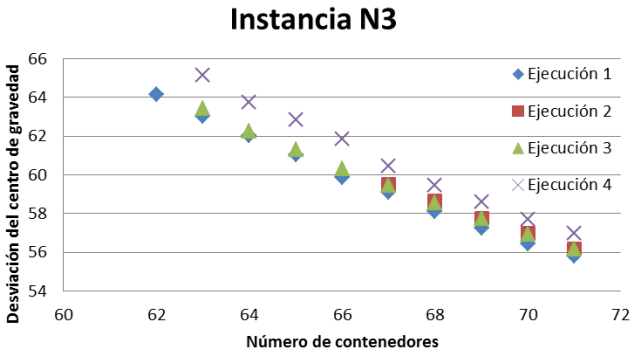


**Figura 22.** Comparativa de frentes entre distintas ejecuciones.

| **Parámetros Fijos** | **%** |
|---|---|
| Probabilidad de aplicar Mutación | 10 |
| *Probabilidad de ejecutar la Mutación 1* | *25* |
| *Probabilidad de ejecutar la Mutación 2* | *25* |
| *Probabilidad de ejecutar la Mutación 3* | *50* |
| Probabilidad de Cruce | 50 |
| Probabilidad del Optimizador | 70 |

**Tabla 2.** Parámetros fijos establecidos.

Como puede observarse, la probabilidad de mutación tiene un valor pequeño de un 10%, ya que la mutación es un factor que suele darse poco en el contexto de la genética. Al contrario, el operador de cruce alcanza un valor de un 50% de probabilidad, ya que es una condición que se cumple con facilidad en la naturaleza respecto a la mutación. El optimizador es el encargado de explorar el espacio de soluciones intentando mejorar las soluciones en cada iteración, por lo cual le daremos un valor alto del 70% para intentar mejorar las soluciones. Por último, en cuanto al proceso de selección, en cada generacion se almacenará en un archivo externo el frente de Pareto obtenido, sustituyendo este frente por las soluciones que peor objetivo 1 tengan, al ser estas las soluciones más factibles en la vida real, los que necesitan menor número de contenedores.

## 6.2 Métricas utilizadas en el algoritmo multi-objetivo

Una de las métricas más utilizadas para comparar distintos frentes de Pareto es la métrica de cobertura [61]. La cobertura *C(A,B)* calcula el porcentaje de soluciones del conjunto de soluciones no dominadas *B* dominadas por al menos una solución del conjunto de soluciones no dominadas *A*.

$$C(A,B) = \frac{\left|\{b \in B \mid \exists a \in A : a \prec b\}\right|}{|B|} \quad (5)$$

Por otro lado la métrica de hipervolumen fue introducida por Zitzlery y col. [62]. El hipervolumen mide el volumen de cada solución no dominada respecto al espacio de los objetivos. Geométricamente hablando, el indicador de hipervolumen mide el volumen del espacio dominado por todas las soluciones contenidas en el conjunto $X \subseteq \Re^d$. Este espacio es calculado respecto a un punto de referencia $r=(r_1,r_2,...,r_d)$. El hipervolumen HYP(X) de un conjunto de soluciones *X* se define como:

$$HYP(X) := VOL\left(\bigcup_{(x_1,...,x_d)\in X} [r_1,x_1]\times...\times[r_d,x_d]\right) \quad (6)$$

## 6.3 Resultados para el algoritmo evolutivo multi-objetivo

Debido a la escasa literatura al respecto de este problema, hemos generado nuestras propias instancias para probar el buen funcionamiento del algoritmo. Las instancias han sido generadas con distintos tamaños para estudiar cómo trabaja el algoritmo evolutivo cuando las instancias son de gran tamaño. El tamaño de los contenedores se fija para todos a un tamaño de 13,5 metros de longitud por 2,4 metros de ancho, que son las medidas estándares de los remolques de los trailers actualmente. Partiendo de que vamos a usa los palets europeos al ser estos los más utilizados en la Unión Europea, las medidas para todos los palets será de 1,2 metros de largo por 0,8 metros de ancho, de esta forma el máximo número de palets que podremos insertar en cada camión será de 33 palets. Por último se ha establecido una carga máxima de cada camión de 25000 kg, no pudiéndose superar este a la hora de cargar los palets Una vez establecidos estos parámetros, genera un conjunto de instancias variando el número de palets, el peso de los mismos y el número de clientes a los que hay que enviar estos palets. Cada palet podrá estar asociado a solo un cliente. El anexo 1 muestra una posible configuración de una instancia. A la hora de generar los pesos de los palets se han generado de forma aleatoria acorde con los pesos medios de los palets cargados con productos agrícolas. Estos pesos se generaran para cada palet con valores que oscilan entre los 600 y 900 kg dependiendo del producto que se cargue. Como restricción recordar que los palets de un cliente deben de llevarse en el mismo camión por facilidades logísticas a la hora del transporte. La tabla 3 muestra los tamaños de las distintas instancias generadas.

| Instancia | Número de palets | Número de Clientes |
|---|---|---|
| N1 | 500 | 50 |
| N2 | 1000 | 100 |
| N3 | 2000 | 200 |
| N4 | 5000 | 500 |
| N5 | 7500 | 750 |
| M1 | 2500 | 150 |
| M2 | 5000 | 400 |
| M3 | 7500 | 550 |

**Tabla 3.** Conjunto de instancias utilizadas.

El algoritmo evolutivo implementado TP-MOEA tiene un gran número de parámetros de entrada, como hemos comentado anteriormente, los cuales han sido modificados para obtener diferentes resultados, quedando otro conjunto de parámetros fijos. A la hora de ejecutar el algoritmo, se han utilizado tres valores distintos para el tamaño de la población (individuos de la población, $P_{size}$) los cuales han sido fijados como: $P_{size}$={50, 100, 200}. En cuanto al criterio de parada, se ha optado por ejecutar el algoritmo durante 25,000 evaluaciones.

Para establecer un valor de población correcta se han ejecutado las instancias de tipo N para distintos tamaños de población comparando los resultados obtenidos de cada frente con las métricas multiobjetivo explicadas anteriormente. La Figura 23 muestra los frentes de soluciones para la ejecución de las instancias de tipo N, como se puede apreciar, el resultado obtenido es un frente de Pareto con las mejores soluciones obtenidas. Para cada instancia del problema, obtenemos un frente de Pareto con el conjunto de soluciones no dominadas que nos permite elegir entre distintas posibilidades a la hora de empaquetar los palets en los camiones. En el caso de la instancia N1 para un tamaño de población de 100 individuos podemos optar por insertar los palets en 16 camiones con un balanceo de carga de 62 o insertar los palets en 18 camiones consiguiendo reducir dicho balanceo de carga a un valor de 53,3. De esta forma será el usuario final el encargado de elegir la solución que más se adapte a sus necesidades.

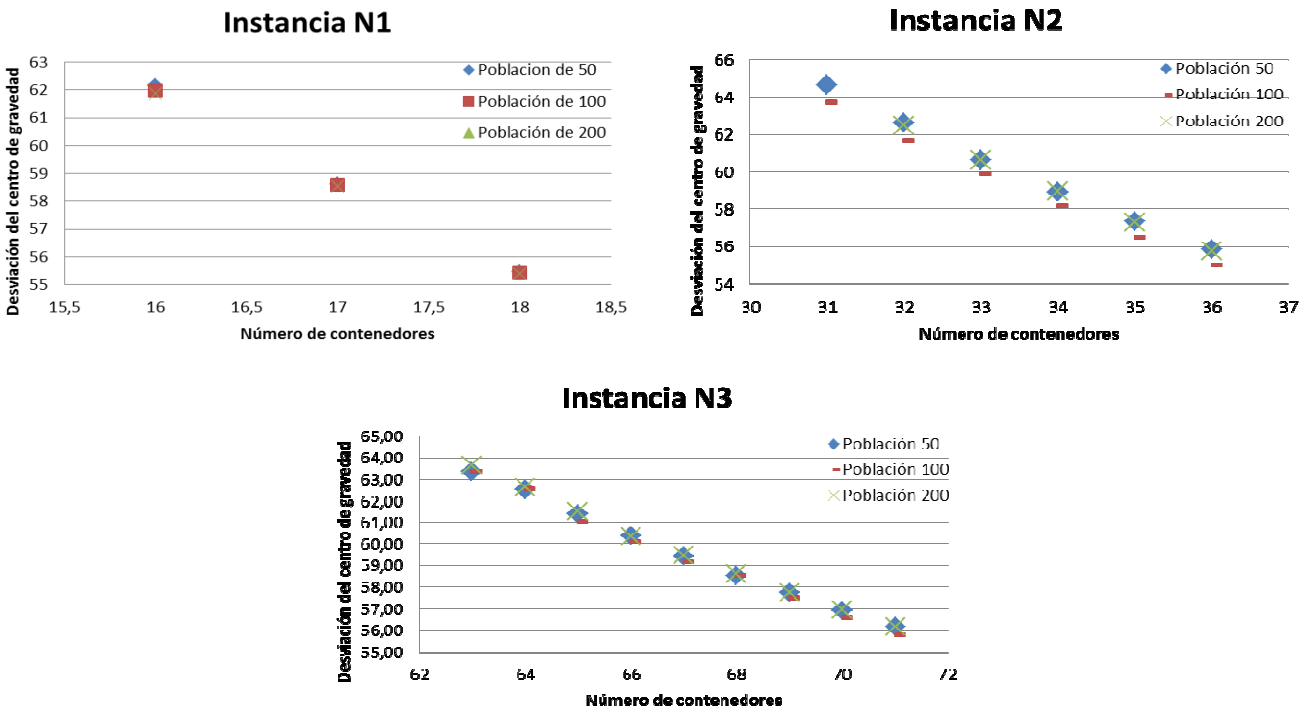


**Figura 23.** Frentes de Pareto para distintos tamaños de población.

Como se puede apreciar en las soluciones obtenidas para las instancias ejemplo que tienen menor número de palets los frentes de Pareto obtenidos tienen menor número de soluciones. Conforme aumenta la dificultad del problema y las instancias son más grandes, los frentes son de mayor número de soluciones no dominadas. Visualmente, es difícil comparar los resultados obtenidos al tener un conjunto de soluciones. Para ello los resultados han sido comparados con las métricas más utilizadas en la optimización multi-objetivo. Como se puede apreciar en los frentes, el tamaño de población que obtiene siempre los mejores resultados es el de 100 individuos. Las tablas 4 y 5 muestran los resultados de comparar los frentes de soluciones de distintas instancias para tres tamaños de población.

.

| **Instancia** | | **N1** | | | **N2** | | | **N3** | |
|---|---|---|---|---|---|---|---|---|---|
| Población | 50 | 100 | 200 | 50 | 100 | 200 | 50 | 100 | 200 |
| 50 | - | 0 | 0 | - | 0 | 0.4 | - | 0.22 | 0.77 |
| 100 | 1 | - | 0 | 1 | - | 1 | 0.77 | - | 1 |
| 200 | 1 | 1 | - | 0,5 | 0 | - | 0.22 | 0 | - |

**Tabla 4**. Comparativa de frentes de Pareto en términos de cobertura.

| **N1** | | | **N2** | | | **N3** | | |
|---|---|---|---|---|---|---|---|---|
| 50 | 100 | 200 | 50 | 100 | 200 | 50 | 100 | 200 |
| 18.57 | 18.80 | **18.98** | 56.87 | **62.71** | 53.91 | 92.01 | **95.18** | 91.38 |

**Tabla 5.** Comparativa de frentes de Pareto en términos de hipervolumen.

Como se aprecia en los resultados calculados tanto para la métrica de cobertura como la de hipervolumen las soluciones obtenidas para tamaños de población de 100 individuos mejoran a los resultados obtenidos por las demás ejecuciones.

Con los parámetros utilizados, el algoritmo evolutivo alcanza resultados de gran calidad, superando todas las restricciones analizadas inicialmente. Para instancias de pequeño tamaño como N1 las restricciones son más fáciles de cumplir y obtenemos frentes de Pareto con menor número de soluciones.

Por último, destacar la importancia del trabajo que realiza el optimizador, el cual trata de mejorar el número de contenedores necesarios para albergar todos los palets, buscando los huecos libres

y maximizando el espacio. La figura 24 TP-MOEA muestra un ejemplo de ejecución del algoritmo evolutivo para la instancia M3 con el optimizador y sin el mismo, donde se aprecia como el optimizador consigue llegar a soluciones con menor número de contenedores que sin su uso. Esto se debe a que en el caso de no usar el optimizador, el algoritmo se centra en minimizar el balanceo de carga de los contenedores, pero no consigue ir reduciendo el número de contenedores.

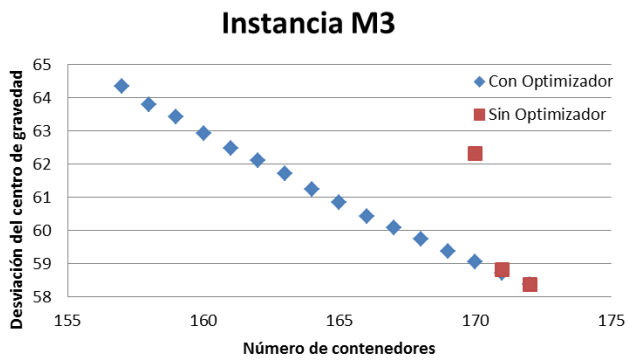


**Figura 24.** Comparativa del uso del optimizador.

## 6.4 Resultados experimentales para el algoritmo evolutivo multi-objetivo paralelo basado en islas PTP-MOEA

La versión paralela ha sido implementada basándonos en un modelo de islas. El modelo se ha ejecutado utilizando cuatro islas, ya que al incrementar el número de islas a ocho o más no mejoraban los resultados. La tabla 6 muestra los parámetros usados para los diferentes operadores evolutivos. Los parámetros han sido establecidos como muestra la tabla tras una serie de ejecuciones con distintos parámetros, hasta comprobar los valores más eficientes para los distintos operadores evolutivos.

| Operador | Isla 1 | Isla 2 | Isla 3 | Isla 4 |
|---|---|---|---|---|
| Mutación 1 (%) | 10 | 10 | 10 | 10 |
| Mutación 2 (%) | 15 | 15 | 15 | 15 |
| Mutación 3 (%) | 25 | 25 | 25 | 25 |
| Cruce (%) | 20 | 20 | 20 | 20 |
| Optimizador (%) | 35 | 55 | 55 | 55 |
| Frecuencia de Migración (iteraciones) | 200 | | | |
| Individuos en cada isla | 50 | | | |

**Tabla 6.** Configuración de parámetros para cada isla.

Al aplicar en cada isla diferentes probabilidades de mutación y el optimizador, se pretende que cada isla intente mejorar objetivos distintos, explorando así espacios de búsqueda diferentes. El mecanismo de elitismo consiste en un proceso de migración. Cada 200 iteraciones las islas mandan las soluciones no dominadas del frente de Pareto a sus vecinos, los cuales reemplazan sus soluciones dominadas.

El criterio de reemplazo de las soluciones dominadas son las soluciones con peor objetivo 1, es decir las soluciones con mayor número de contenedores. El criterio de parada es de 25000 evaluaciones, lo mismo que en la versión secuencial. La figura 25 muestra el frente de Pareto generado por el algoritmo paralelo usando cuatro islas y el frente obtenido por la versión secuencial.

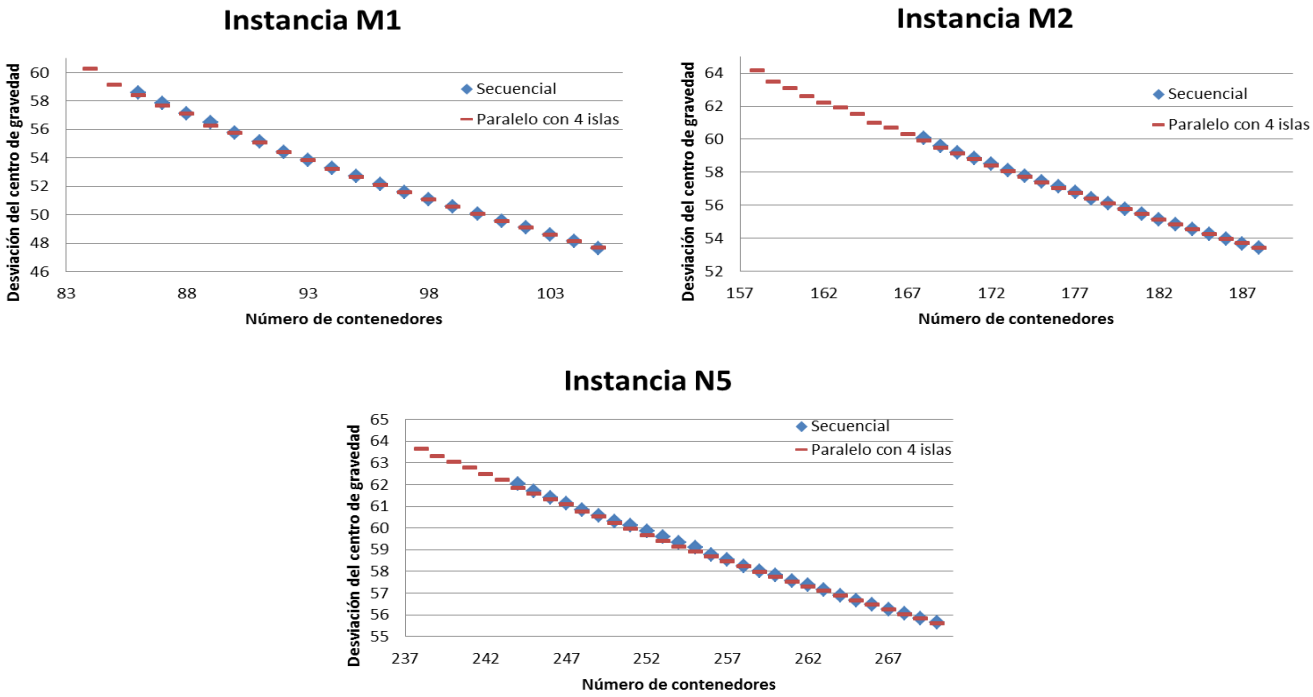


**Figura 25.** Frentes de Pareto para distintos tamaños de población.

Los dos objetivos mostrados entran en conflicto y por tanto se incrementan el número de contenedores necesarios para albergar todas las piezas pero el equilibrio de la carga entre los diferentes contenedores mejora.

Se puede observar en la figura 25 como el algoritmo paralelo consigue obtener soluciones con menor número de contenedores al conseguir profundizar en el mismo número de evaluaciones en un espacio de búsqueda mayor.

Las tablas 7 y 8 muestran los resultados obtenidos con PTP-MOEA en términos de cobertura e hipervolumen, y permite concluir que la versión paralela alcanza mejores resultados que la versión secuencial en el mismo número de evaluaciones. Al ser problemas de complejidad NP-Duro, en los cuales el tiempo de cómputo es muy elevado, la implementación de algoritmos paralelos se hace necesario en ciertos problemas.

| **Instancia** | **M1** | | **M2** | | **N5** | |
|---|---|---|---|---|---|---|
| Algoritmo | TP-MOEA | PTP-MOEA | TP-MOEA | PTP-MOEA | TP-MOEA | PTP-MOEA |
| TP-MOEA | - | 0.22 | - | 0.25 | - | 0.09 |
| PTP-MOEA | 0.77 | - | 0.61 | - | 0.88 | - |

**Tabla 7.** Comparativa de frentes de Pareto en términos de cobertura.

| **M1** | | **M2** | | **N5** | |
|---|---|---|---|---|---|
| TP-MOEA | PTP-MOEA | TP-MOEA | PTP-MOEA | TP-MOEA | PTP-MOEA |
| 278.81 | **286.54** | 345.89 | **399.18** | 360.93 | **386.36** |

**Tabla 8**. Comparativa de frentes de Pareto en términos de hipervolumen.

Para conocer el buen funcionamiento del algoritmo paralelo es importante conocer el speed-up del mismo. El speed-up para *p* procesadores, *Sp*, es el cociente entre el tiempo de ejecución de un programa secuencial, *TS*, y el tiempo de ejecución de la versión paralela de dicho programa en *p* procesadores, *TP*. Dado que puede haber distintas versiones secuenciales, se elige el *TS* de la versión secuencial más rápida. Este ´índice indica la ganancia de velocidad que se ha obtenido con la ejecución en paralelo.

$$S_p = \frac{T_s}{T_p} \tag{7}$$

Por ejemplo, un speed-up igual a 2 indica que se ha reducido el tiempo a la mitad al ejecutar el programa con varios procesadores. En el mejor de los casos el tiempo de ejecución de un programa en paralelo con *p* procesadores será *p* veces inferior al de su ejecución en un sólo procesador, teniendo todos los procesadores igual potencia de cálculo. Es por ello que el

valor máximo que puede alcanzar el speed-up de un algoritmo paralelo es *p*. Generalmente el tiempo no se verá reducido en un orden igual a *p*, ya que hay que contar con la sobrecarga extra que aparece al resolver el problema en varios procesadores, debido a sincronizaciones y dependencias entre ellos.

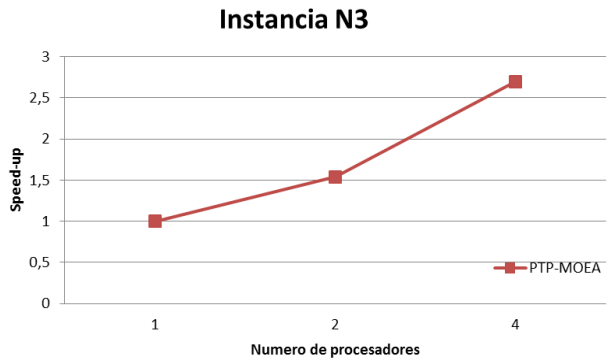


**Figura 26.** Speed-up obtenido por PTP-MOEA para 1,2 y 4 procesadores.

## 7. Conclusiones

El transporte de palets en camiones se puede clasificar como un problema de empaquetamiento multi-objetivo de tipo NP-Hard, es por esto que se ha implementado un algoritmo evolutivo multi-objetivo TP-MOEA para resolver este problema. Se han diseñado nuevos operadores evolutivos adaptados al problema del almacenamiento de palets en camiones, para el buen funcionamiento del algoritmo secuencial, los cuales han obtenido resultados de gran calidad. Entre los operadores implementados destaca un optimizador, que consigue obtener soluciones con un número mínimo de contenedores. También se ha implementado una versión paralela PTP-MOEA basada en un modelo de islas para mejorar el rendimiento del algoritmo. Debido a la escasa o nula literatura encontrada que trate este problema de forma multi-objetivo, se diseñaron un conjunto de benchmarks de distintos tamaños con medidas reales obtenidas de empresas de transportes. Estos benchmarks han sido ejecutados con ambos algoritmos y comparados los resultados con distintas métricas, corroborando la eficiencia del algoritmo paralelo respecto a su versión secuencial. Los resultados obtenidos para cada instancia están formados por un conjunto de soluciones donde el cliente puede elegir entre soluciones donde se aprovecha al máximo el espacio de los camiones, con soluciones donde se desperdicia el espacio en beneficio de un balanceo de carga de estos, lo cual repercute beneficiosamente en el ahorro de combustible y otros factores. Los resultados obtenidos demuestran el buen funcionamiento de los algoritmos evolutivos cuando trabajamos con problemas que son difíciles de solucionar con técnicas deterministas. Como futuro trabajo se puede integrar el problema de empaquetado de palets con otros problemas de transporte, como es el problema del viajante de comercio. O el uso de otras meta-heurísticas para la resolución de estos tipos de problemas.

## 8. Referencias


[1] **Dowsland** K.A., Dowsland W.B., "Packing problems", European Journal of Operational Research56, 2–14, 1992.

[2] **Dyckhoff** H., Scheithauer G., Terno J., "Cutting and packing (C&P)", in: M. Dell'Amico, F. Maffioli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, Chichester, 393–413, 1997.

[3] **Dyckhoff** H., Finke U., "Cutting and Packing in Production and Distribution", Physica Verlag, Heidelberg, 1992.

[4] **Coffman Jr.** E.G., Shor P.W., "Average-case analysis of cutting and packing in two dimensions", European Journal of Operational Research44, 134–144, 1990.

[5] **Coffman Jr.** E.G., Lueker G.S., "Probabilistic Analysis of Packing and Partitioning Algorithms", Wiley, Chichester, 1992.

[6] **Liu**, D., Tan, K., Huang, C. and Ho, W,. "On solving multiobjective bin packing problems using evolutionary particle swarm optimization", European Journal of Operational Research, 190(2), 357–382, 2008.

[7] **Gilmore** P.C., Gomory R.E., "Multistage cutting problems of two and more dimensions", Operations Research13, 94–119, 1965.

[8] **Directorio Central de Empresas: explotación estadística** disponible en: http://www.ine.es/jaxi/menu.do?type=pcaxis&path=/t37/p201/&file=inebase.

[9] **ITF2008**. Transport and energy: the challenge of climate change – research findings. In: International Research Forum, Leipzig, OECD/ITF, disponible en: http://www.internationaltransportforum.org/.

[10] **Crainic** T.G., "A survey of optimization models for long-haul freight transportation", Handbook of Transportation Science, R.W. Hall (Ed.), second edition, Kluwer, 2002.

[11] **Siri** S., "Modelling, optimization and control of logistic systems", PhD Dissertation, University of Genova.

[12] **Bektaş** T., Crainic T.G., "A brief overview of intermodal transportation, Logistics Engineering Handbook", G.D. Taylor (Ed.), Taylor and Francis Group, 2007.

[13] **Garey** M.R., Johnson D.S., "Computers and intractability: a guide to the theory of NP-completeness", San Francisco, CA: W.H. Freeman & Company, 1979.

[14] **Ong** H.L., Magazine M.J., and Wee T.S., "Probabilistic Analysis of Bin Packing Heuristics", Operations Research, 32(5), 983-998, 1984.

[15] **Hopper** E., and Turton B., "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem", European Journal of Operational Research, 128, 34-57, 2001.

[16] **Sathe** M., Schenk O., and Burkhart H., "Solving bi-objective manyconstraint bin packing problems in automobile sheet metal forming processes", EMO '09: Proceedings of the 5th International Conference on

Evolutionary Multi-Criterion Optimization, 246-260, Berlin, Heidelberg, 2009.

[17] **Geiger** M. J., "Bin packing under multiple objectives — a heuristic approximation approach", In The Fourth International Conference on Evolutionary Multi-Criterion Optimization: Late Breaking Papers, 53- 56, Matsushima, Japan, 2007.

[18] **Goldberg** D.E., "Genetic Algorithms in Search Optimization and Machine Learning", Addison-Wesley-Longman, Reading, Mass.1989.

[19] **Baños** R., "Meta-heurísticas híbridas para optimización mono-objetivo y multi-objetivo: Paralelización y aplicaciones", Tesis Doctoral, Editorial Universidad de Almería, ISBN: 978-84-8240-878-1, 2006.

[20] **Jacobson** S.H., Yucesan E., "Analyzing the Performance of Generalized Hill Climbing Algorithms", Journal of Heuristics 10(4), 387-405.

[21] **Kirkpatrick** S., Gelatt C.D., Vecchi M.P., "Optimization by Simulated Annealing", Science 220, 671, 1983.

[22] **Gil** C., Ortega J., Montoya M.G., Baños R., "A Mixed Heuristic for Circuit Partitioning", Computational Optimization and Applications Journal 23(3), 321-340, 2002.

[23] **Glover** F., Laguna M., Dowsland K.A., "Modern Heuristic Techniques for Combinatorial Problems", C.R. Reeves (eds.), Blackwell, London, 1993.

[24] **Coello** Coello, C.A., "Introducción a la Computación Evolutiva (Notas de Curso)", 67, Abril 2006

[25] **Hoffman** A., "Arguments on Evolution: A Paleontologist's Perspective", Oxford University Press, New York, 1989.

[26] **Fogel** D. B., "Evolutionary Computation. Toward a New Philosophy of Machine Intelligence". The Institute of Electrical and Electronic Engineers, New York, 1995.

[27] **Glover** F., "Heuristics for Integer Programming using Surrogate Constraints", Decision Sciences 8, vol. 1, 555-568, 1997.

[28] **Laguna** M., Marti R., "Scatter Search Methodology and Implementations in C", Kluwer Academic Publishers, Boston, 2003.

[29] **Feo** T., Resende M., "Greedy Randomized Adaptive Search Procedures", Journal of Global Optimization 6, 109-133, 1995.

[30] **Resende** M., González-Velarde J.L., "GRASP: Procedimientos de Búsquedas Miopes Aleatorizados y Adaptativos", Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 19, 61-76, 2003.

[31] **Hansen** P., Mladenovic N., "An Introduction to Variable Neighborhood Search", Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, S. Voss et al. (eds.), Kluwer Academic Publishers, Boston, 1999.

[32] **Dorigo** M., Maniezzo V. and Colorni A., "Ant System: Optimization by a colony of cooperating agents". IEEE Transactions on Systems, Man and Cybernetics - Part B, 26(1): 29-41, 1996.

[33] **Dawkins** R., "The selfish gene", Oxford University Press, New York, 1976.

[34] **Moscato** P.: "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms". California Institute of Technology Technical Report C3P 826, Pasadena, CA 1989.

[35] **Moscato**, P., "Memetic algorithms: A short introduction", New Ideas in Optimization, McGraw-Hill, 219–234, 1999

[36] **Mohamed-Ahemed** B., Yassine A., "Optimization by ant colony hybrid for the bin-packing problem". World Academy of Science, Engineering and Technology, 49, 354-357, 2009.

[37] **Reynolds** R.G., Michalewicz Z., Cavaretta, M., "Using cultural algorithms for constraint handling in GENOCOP". In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, Proceedings of the Fourth Annual Conference on Evolutionary Programming, 298–305, 1995.

[38] **Kennedy** J., Eberhart R., "Particle swarm optimization", in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, 1942–1948, 1995.

[39] **Fonseca** C.M., **Fleming** P.J., "Genetic Algorithms for Multiobjective Optimization: Formulation", Discusión and Generalization, Conf. on Genetic Algorithms, San Mateo, CA, 1993.

[40] **Coello** C., "A. Optimización evolutiva con objetivos múltiples: Estado del arte y tendencias futuras", Engineering design centre, Plymouth university, UK. 1998.

[41] **Goldberg** D. E., "Genetic algorithms in search, optimization and machine learning". New York, Addison Wesley, 1989.

[42] **Fonseca** C.M., Fleming P.J., "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization", In Proceedings of the Fifth International Conference on Genetic Algorithms, 416-423, 1993.

[43] **Srinivas** N., Deb K., "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms", Evolutionary Computation, 2(3), 221-248, 1994.

[44] **Deb** K., Pratap A., Agrawal S. and Meyarivan T., "A fast and elitist multiobjective genetic algorithm: NSGA-II" Technical Report No. 2000001. Kanpur: Indian Institute of Technology Kanpur, India, 2000.

[45] **Horn** J., Nafpliotis N., Goldberg D., "A Niched Pareto Genetic Algorithm for Multiobjective Optimization", In Proceedings of the First IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA, 82-87, 1994.

[46] **Zitzler** E., Thiele L., *"Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach",* IEEE Transactions on Evolutionary Computation 3(4), 257-271, 1999.

[47] **Zitzler** E., Laumanns M., Thiele L., "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization*"*, In Proc. of Evolutionary Methods for Design, Optimisation, and Control, Barcelona, Spain, 95-100, 2001.

[48] **Corne** D.W., Knowles J.D., Oates, M.J., "The Pareto-envelope Based Selection Algorithm for Multiobjective Optimization", In Proceedings of Parallel Problem Solving from Nature VI, Schoenauer y col. (eds), Springer-Verlag, 839-848, 2000.

[49] **Corne** D.W., Jerran N.R., Knowles J.D., Oates M.J., "PESA-II: Region-based Selection in Evolutionary Multi-objective Optimization", Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann Publishers, 283-290, 2001.

[50] **Krishnakumar** K., "Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization", Intelligent Control and Adaptive Systems, 1989.

[51] **Toscano** G., Coello C.A. "The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization", Lecture Notes in Computer Science 2632, 252-266, 2003.

[52] **Baños** R., Gil C., Paechter B., Ortega J., "A Hybrid Meta-heuristic for Multi-objective Optimization: MOSATS", Journal of Mathematical Modelling and Algorithms, 6:2, 213-230, 2007.

[53] **Baños** R., Gil C., Reca J., Martínez J., "Implementation of Scatter Search for Multi-Objective Optimization: A Comparative Study, Computational Optimization and Applications", J. 42, 421:441, 2009.

[54] **Beausoleil** R.P., "MOSS-II Tabu/Scatter Search for Nonlinear Multiobjective Optimization, Advances in Metaheuristic Methods for Hard Optimization", 39-67, Berlin, 2008.

[55] **Schütze** O., Coello C. A., Mostaghim S., Talbi E.-G. and Dellnitz M., "Hybridizing evolutionary strategies with continuation methods for solving multiobjective problems", Eng. Optimization,. 40: 5,. 383–402, May 2008.

[56] **Schütze** O., Sanchez G. and Coello C. A., "A new memetic strategy for the numerical treatment of multiobjective optimization problems," in Proc. Genetic Evol. Comput. Conf. (GECCO '08), Atlanta, GA: ACM, 705–712, Jul. 2008.

[57] **Cantu-Paz** E., "A survey of parallel genetic algorithms," Calculateurs Paralleles, Reseaux et Systemes Repartis, 10:2, 141–171, 1998.

[58] **Van Veldhuizen** D. A., Zydallis J. B., and Lamont G. B., "Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms", IEEE Transactions on evolutionary Computation, 7:2,. 144–173, 2003.

[59] **Alba** E., "Parallel Metaheuristics: A New Class of Algorithms", Wiley, 2005.

[60] **Davidor** Y. and Ben-Kiki O., "The interplay among the genetic algorithm operators: Information theory tools used in a holistic way". In R. Manner and B. Manderick, editors, Parallel Problem Solving From Nature II, 75–84, 1992.

[61] **Zitzler** E., Thiele L., Laumanns M., Fonseca C.M., and Grunert da Fonseca V., "Performance assessment of multiobjective optimizers: an analysis and review", IEEE Transactions on Evolutionary Computation, 7:2, 117-132, 2003.

[62] **Talbi** E. G., "Metaheuristics: From design to implementation", Wiley, 2009.

**Apéndice 1**

```
200             //Número de clientes
13500  2400     //Tamaño del contenedor
800    1200     //Tamaño del palet
25000           // Máximo peso de carga del camión (kg)
33     // Máximo número de palets por camión
1000  //Número de palets

//Peso del palet y número de cliente
655,149
536,5
780,55
…
…
…
823,59
613,173
753,90
900,62
547,200
900,156
745,98
775,27
875,56
873,58
745,87
532,95
685,145
732,17
```

UNIVERSIDAD DE ALMERÍA

## Resumen/Abstract

En este trabajo se ha implementado un algoritmo evolutivo multi-objetivo paralelo, para resolver el problema de empaquetamiento en dos dimensiones con restricciones, para una aplicación de transporte de palets en camiones. El transporte de palets en camiones tiene una gran importancia en Andalucía y especialmente en el campo almeriense donde a diario salen hacia Europa cientos de camiones cargados de productos del campo. El problema de empaquetamiento en dos dimensiones (2DPP) consiste en insertar un conjunto de objetos caracterizados por tener un alto y ancho específico, en el menor número de camiones posibles donde el alto y ancho es igual para todos. A partir de esta definición existen multitud de variantes al problema. En la variante multi-objetivo del problema, además de minimizar el número de camiones, se intenta minimizar el balanceo de carga de los mismos intentando colocar la carga de las piezas de la mejor forma posible para que el centro de gravedad del camión quede lo más cercano posible al centro deseado. El algoritmo se ha aplicado a una variante del problema donde se trata de insertar un conjunto de palets con su alto, ancho y peso específico, en el menor número de camiones posible y con el mejor balanceo de carga, evitando una serie de restricciones añadidas al problema. Cada palet corresponde a un cliente, con lo cual todos los palets de un mismo cliente deben de ir en el mismo camión, cada camión no puede ir cargado con más de 25000 kilos y el centro de gravedad debe de ir lo más próximo al eje del camión. Para la optimización de este problema hemos implementado un algoritmo evolutivo TP-MOEA, este tipo de algoritmos están inspirados en la teoría de la evolución de Darwin y en el desarrollo de la informática evolutiva. Los algoritmos evolutivos son técnicas de optimización y búsqueda de soluciones basadas en la selección natural y genética que permiten resolver problemas no lineales en los que interviene un alto número de variables en problemas complejos. El algoritmo ha sido implementado con un conjunto de operadores evolutivos diseñados para obtener soluciones de gran calidad para un conjunto de instancias establecidas.