Universidad de Valladolid

FACULTAD DE CIENCIAS

GRADO EN ESTADÍSTICA

ESTUDIO DE MÉTODOS EXACTOS Y APROXIMADOS PARA LA RESOLUCIÓN DEL PROBLEMA DE LOCALIZACIÓN SIN CAPACIDADES

Alumno:
D. Gabriel Rodríguez Canal

Tutores:
Dr. Jesús Alberto Tapia García
Dr. Jesús Sáez Aguado

A mi familia.

Agradecimientos

Quiero agradecer a mi tutor, Jesús Sáez, su implicación durante este trabajo y todos sus consejos para poder llevarlo a buen puerto. Gracias a su guía he podido profundizar en un área de la Estadística que considero interesante y adquirir conocimientos útiles para mi futura carrera profesional.

Me gustaría mencionar también a mi familia, que me ha dado su apoyo emocional durante el desarrollo de este trabajo.

Resumen

Este trabajo estudia el problema UFLP (problema de localización sin capacidades), que constituye la base de los problemas de localización. Dado que se trata de un problema computacionalmente costoso los métodos exactos no son especialmente útiles para tamaños grandes del problema.

El foco del trabajo se encuentra en las heurísticas de búsqueda local y las metaheurísticas, que permiten resolver problemas de optimización grandes en tiempos practicables. Se estudian los algoritmos mediante su implementación en C y la resolución de ficheros de datos de diferentes características y se comparan con los métodos exactos. Finalmente, se determina qué metaheurística es más apropiada para la resolución de este problema.

Abstract

This work studies the UFLP (Uncapacitated Facility Location Problem), wich lays the basis of localisation problems. Since this is a computationally costly problem the exact methods are not particularly useful for big instances of the problem.

The focus of this work is on local search heuristics and metaheuristics, which enable solving big optimisation problems in reasonable times. Different algorithms are studied through their implementation in C and the solution of datasets of diverse features and they are compared to the exact methods. Finally, it is determined which metaheuristic is the most adequate for the solution of this problem.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Estructura del trabajo

Este trabajo aborda el problema UFLP (*Uncapacitated Facility Location Problem*). En el capítulo 2 se describen el problema y sus aplicaciones. Se trata de un problema de programación entera. Existen dos formas de resolver estos problemas: exacta y aproximada. En el capítulo 3 se describen los métodos de resolución exacta utilizados en este trabajo. Aunque garantizan la obtención del valor objetivo óptimo, en la práctica esto solo se da para problemas pequeños, dado que, debido a la complejidad del problema, las instancias más grandes no pueden resolverse en un tiempo practicable. Es por ello que surgen formas aproximadas de resolverlos. El resto del trabajo se enfoca en este tipo de métodos. En el capítulo 4 se presentan las heurísticas de mejora diseñadas para este problema, según la literatura. Estas heurísticas se enmarcan dentro del paradigma *greedy*, por lo que no garantizan la obtención del óptimo, pero proveen una primera solución que, en muchos casos, puede ser de una calidad aceptable. A partir de esta solución se puede realizar una búsqueda local, tal y como se describe en el capítulo 5, que tratará de mejorar la solución inicial realizando una búsqueda en el entorno de la misma. Con el objetivo de aumentar la probabilidad de encontrar una solución lo más cercana al óptimo posible se emplean las metaheurísticas, introducidas en el capítulo 6. Estos algoritmos alternan entre búsqueda global (diversificación) y búsqueda local (intensificación), de modo que necesitan de un procedimiento de búsqueda local, como los descritos en el capítulo anterior.

1.2. Metodología de experimentación

La experimentación se realizó en una máquina con procesador Intel Core i7-4790 @ 3.60GHz, con 16 GB de RAM y sistema operativo Windows 10 x64. Para los métodos de resolución exacta se empleó el *solver* Xpress Mosel. La implementación de los métodos de resolución aproximada se realizó en el lenguaje de programación C. Los *scripts* de lanzamiento

por lotes de los *benchmarks* de los métodos aproximados, así como los de post-tratamiento de los datos, se escribieron en Python.

Los resultados de los métodos exactos se recogen en la tabla A.1. Esta tabla ofrece, además, el índice asociado a cada fichero y el tamaño del mismo, por lo que debería usarse como referencia durante la lectura del trabajo. Los índices se utilizan en los gráficos de tiempos de solución de las metaheurística, como 6.2.

A lo largo del trabajo se emplea la expresión *índice de optimalidad* para los análisis de resultados. Se trata de una expresión *ad hoc* que expresa:

$$\frac{\text{nº de ficheros en los que se alcanzó el óptimo}}{\text{nº total de ficheros}}$$

1.2.1. Ficheros empleados

En este trabajo se han empleado ficheros de tres fuentes diferentes, que con frecuencia aparecen en la literatura.

- UnifS: estos ficheros tienen dimensión n = m = 100 en todos los casos. El coste fijo de abrir una instalación es de 3000, mientras que el coste de transporte proviene de una distribución uniforme [0, 10000]. Fueron generados por Kochetov e Ivanenko (Kochetov e Ivanenko, 2005). Pueden encontrarse en la página web del Instituto de Matemáticas de Sobolev.
- M*: estos ficheros fueron intencionalmente diseñados para parecerse a los problemas reales, con un gran número de soluciones cercanas al óptimo. Hay 6 clases de problemas y en todos ellos m = n. El paquete utilizado en este trabajo fue generado por Laurent Michel y fue obtenido de la página web del Instituto de Informática Max Planck.
- cap: se trata de una colección de instancias disponibles en ORLIB. Probablemente se trate de los ficheros más usados para evaluar el rendimiento de algoritmos de resolución de UFLP. Pueden obtenerse en la página web del Instituto de Informática Max Planck.

1.2.2. Métodos exactos

Se han estudiado las formulaciones débil y fuerte para el problema UFLP, que han sido resueltas mediante el método *Branch and Bound*, el método de los planos de corte y el de *Branch and Cut*. Para aplicar el algoritmo *Branch and Bound* se ha utilizado el *solver* Xpress Mosel, con un tiempo límite de ejecución para la fase de resolución del problema entero de 60 segundos (en la tabla A.1 se puede observar que para algunas instancias se supera este umbral de tiempo. Esto es debido a que la fase de relajación lineal toma más tiempo del límite especificado). Los métodos de planos de corte y *Branch and Cut* se programaron con ayuda de las funciones predefinidas de Xpress Mosel. Al tratarse de métodos deterministas, cada uno se ejecutó una sola vez. Los resultados se muestran en la tabla A.1. En ella aparecen los siguientes campos:

- Índice: cada fichero tiene asociado un índice que se emplea en los gráficos de tiempos de solución de las metaheurísticas en el eje de abscisas.
- Datos: nombre del fichero sobre el que se presentan resultados.
- Tamaño: dimensiones del problema ($n \times m$).
- Método: algoritmo empleado: **weak** (formulación débil), **strong** (formulación fuerte); ambas resueltas mediante *Branch and Bound*, **CP** (planos de corte) y **B&C** (*Branch and Cut*).
- Z_LP: valor objetivo de la relajación lineal del problema.
- nº p.c.: número de planos de corte en el nodo raíz (solo aplica para el método de planos de corte).
- nº var.: número de variables del problema.
- nº rest.: número de restricciones del problema.
- Óptimo: valor objetivo óptimo de la función objetivo. En caso de darse una cota este valor no es el óptimo real.
- Cota Inf.: cota inferior del valor óptimo objetivo. Tan solo se dan cotas en aquellos problemas en los que no se pudo encontrar el óptimo.
- Cota Sup.: cota superior del valor óptimo objetivo.
- TE: tiempo de finalización del método.

1.2.3. Métodos aproximados

Los métodos aproximados fueron reunidos en un programa C para una invocación fácil desde línea de comandos. Junto a este trabajo se proporciona tanto la versión para Windows (*win_uflp-heur.exe*) como la versión para Linux (*uflp-heur*).

Dado que las heurísticas de mejora ADD y DROP siempre llegan a la misma solución, pues son algoritmos deterministas, se ejecutaron estos algoritmos en primer lugar y se almacenaron las soluciones en ficheros. En el capítulo 4 puede encontrarse la tabla asociada (4.1). Cuando se realiza un *benchmark* de una metaheurística para un fichero determinado, se recupera la solución de la heurística de mejora calculada previamente. Esta es suministrada al procedimiento de la metaheurística, que trata de mejorarla hasta alcanzar la condición de terminación. De este modo se evita incurrir en una sobrecarga en términos de tiempo innecesaria.

Los *benchmarks* de las metaheurísticas constan de 10 iteraciones por cada parámetro y fichero. En las tablas de resultados de los *benchmarks*, e.g. la tabla 6.2 de Simulated Annealing, se muestra el rendimiento de un fichero por cada línea. Se indica si el valor objetivo

alcanzado fue óptimo (*****), subóptimo (#) o se mejoró el óptimo teórico (**!**) [1], el mejor valor objetivo alcanzado para el fichero entre todos los parámetros (**BST**), el mejor tiempo de los tiempos medios de solución de las ejecuciones que alcanzaron el óptimo por parámetro (**BST(TS)**) y el mejor tiempo de los tiempos medios de finalización de las ejecuciones que alcanzaron el óptimo por parámetro (**BST(TE)**). Así mismo, se señalan con ✓los parámetros que alcanzaron el valor objetivo **BST** y con ✗los que no.

[1] El valor objetivo óptimo usado para la comparación no necesariamente es el óptimo más actual, sino que se toma el valor suministrado con el conjunto de datos. Así, por ejemplo, será frecuente ver cómo se mejora el óptimo teórico para el fichero MT1 a lo largo del trabajo, con el valor 10069.80. Sin embargo, este valor ya fue alcanzado con anterioridad en otros trabajos. Véase por ejemplo (Michel y Van Hentenryck, 2004).

Capítulo 2

El problema de localización de instalaciones sin capacidad (UFLP)

2.1. Descripción del problema

El problema UFLP (Uncapacitated Facility Location Problem) trata la situación en la que se dispone de n localizaciones en las que se pueden abrir instalaciones de capacidad ilimitada y m clientes. Abrir la instalación j tiene un coste fijo, f_j, y servir al cliente i desde la instalación j tiene un coste de transporte c_{ij}. Los costos c_{ij} pueden intrepretarse también como el costo de asignar **toda** la demanda del cliente i a la instalación j, debido a que UFLP es un problema sin capacidades, lo que se traduce en una capacidad virtualmente infinita en cada instalación. Para determinar qué instalaciones se abren es preciso encontrar un equilibrio entre los costos fijos f_j y los costos de transporte c_{ij}, tal y como se refleja en la función objetivo 2.1.

Las variables binarias x_j representan qué instalaciones se abren ($x_j = 1$ si la instalación j se abre, 0 c.c.). Las variables reales no negativas y_{ij} representan que la instalación j sirve la demanda del cliente i si toman el valor 1.

A continuación se presentan la formulación débil y la formulación fuerte, que comparten algunas restricciones. La formulación débil se caracteriza por tener menos restricciones, por lo que resolver la relajación lineal del problema es más sencillo. Sin embargo, el hueco entre el valor objetivo óptimo de la relajación y el del problema entero es mayor que con la formulación fuerte, por lo que la fase *Branch and Bound* toma más tiempo con la formulación débil. Para UFLP, específicamente, es bien conocido que la formulación fuerte proporciona soluciones enteras o cuasi-enteras y apenas es necesaria la fase de ramificación (*branching*) («Solving facility location models with modern optimization software: the weak and strong formulations

revisited»). En el análisis del capítulo 3 sobre métodos exactos se puede comprobar cómo la formulación fuerte es superior a la débil en este problema.

2.1.1. Formulación débil

La visión tradicional de las formulaciones de los problemas de localización dice que la formulación débil es preferible para problemas complejos. Esta formulación se diferencia de la fuerte en la restricción 2.3, que relaciona las variables y y x estableciendo que si una instalación se abre puede servir puede servir a cualquier número de clientes. En cualquier otro caso, no se le puede asignar ningún cliente.

$$Min \sum_{j \in J} f_j x_j + \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \tag{2.1}$$

$$\sum_{j \in J} y_{ij} = 1, \quad \forall i \in I \tag{2.2}$$

$$\sum_{i \in I} y_{ij} \leq m x_j, \quad \forall j \in J \tag{2.3}$$

$$x_j \in \{0, 1\}, \quad \forall j \in J \tag{2.4}$$

$$y_{ij} \geq 0, \quad \forall i \in I, j \in J \tag{2.5}$$

2.1.2. Formulación fuerte

La formulación fuerte es vista como la preferible para la mayoría de problemas por su capacidad para reducir el hueco de la relajación lineal en relación al óptimo, haciéndola más eficiente para métodos exactos. La diferencia fundamental con respecto a la formulación débil es la restricción 2.8, que indica que cada cliente i solo puede ser servido por una instalación j si está abierta.

$$Min \sum_{j \in J} f_j x_j + \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \tag{2.6}$$

$$\sum_{j \in J} y_{ij} = 1, \quad \forall i \in I \tag{2.7}$$

$$y_{ij} \leq x_j, \quad \forall i \in I, j \in J \tag{2.8}$$

$$x_j \in \{0,1\}, \quad \forall j \in J \tag{2.9}$$

$$y_{ij} \geq 0, \quad \forall i \in I, j \in J \tag{2.10}$$

Donde:

- f_j: costo fijo de abrir la instalación j.
- c_{ij}: costo de servicio desde la instalación j al cliente i.
- x_j: 1 si se abre la instalación j, 0 c.c.
- y_{ij}: 1 si la instalación j sirve al cliente i, 0 c.c.
- $I = \{0, 1, \ldots, n\}$: conjunto de índices de las instalaciones.
- $J = \{0, 1, \ldots, m\}$: conjunto de índices de los clientes.

2.2. Costos

El problema de localización sin capacidades presenta costos fijos f_j asociados a la apertura de una instalación y costos variables c_{ij} función de factores como el transporte. La figura 2.1 muestra cómo se relacionan los dos tipos de costos. Se muestra cómo los costos fijos aumentan de forma lineal, lo que respalda la intuición de que abrir todos los centros posibles no es la mejor estrategia para ahorrar costos. Por otro lado, se ve que los costos de transporte disminuyen a medida que se abren instalaciones. Esto ocurre porque con cada instalación abierta disminuye la distancia a al menos uno de los clientes. Al sumar ambos costos parece claro que es preciso encontrar un equilibrio entre ellos, de modo que se pueda minimizar la función objetivo.

2.3. Aplicaciones

Existen numerosas aplicaciones para el problema de localización sin capacidades, tanto en el sector público como en el privado (Daskin, 2011). El problema de la localización de cuentas bancarias trata de maximizar el número de días que pasa desde que se emite un cheque en un banco j hasta que se cobra una ciudad i con el objetivo de retrasar todo lo posible los pagos de facturas y maximizar el capital disponible en un tiempo dado.

El problema del *clustering* consite en generar particiones de objetos en clases, conocidas como *clusters*, de modo que los elementos dentro del cluster tengan un alto grado de asociación entre ellos y bajo con respecto a los de otros clusters. En el contexto de UFLP este

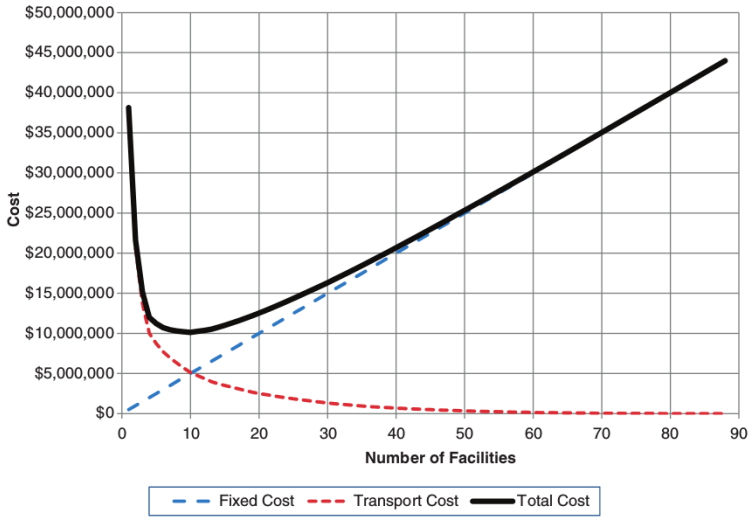

Figura 2.1: Ejemplo de la evolución de los costos fijos, de transporte y totales con el número de instalaciones (Daskin, 2011).

problema se resuelve considerando las instalaciones como individuos promedio del cluster y los clientes como elementos del mismo (Cornuéjols, Nemhauser y Wolsey, 1983).

Finalmente, el problema de localización de cajeros automáticos pretende minimizar el coste de la instalación de estas máquinas sin comprometer la satisfacción de los clientes, tal y como se describe en (Genevois, Turkoglu y Cedolin, 2018).

Capítulo 3

Métodos de resolución exacta

En este capítulo se presentan métodos de resolución exacta para el problema UFLP. La teoría dice que estos métodos son capaces de alcanzar siempre el óptimo global para cualquier problema de programación entera. Sin embargo, en la práctica, tan solo los problemas más pequeños pueden resolverse por estos métodos, ya que los problemas como UFLP son computacionalmente difíciles.

En el apéndice A se encuentra la tabla A.1, que muestra los resultados de la experimentación con los métodos exactos. Allí se pueden encontrar los óptimos para los problemas tratados en el presente trabajo (en aquellos casos en los que fueron tratables). Los problemas más difíciles son los M*, dado que tienen numerosos óptimos locales. En aquellos casos en los que no se haya podido alcanzar el óptimo global real se proporciona una cota, como puede verse en el problema MO1 para la formulación débil, por ejemplo. El valor óptimo no debe tenerse en cuenta.

3.1. Branch and Bound

El problema UFLP es un problema de tipo MIP (*Mixed-integer program*), dado que tiene variables enteras, $\mathbf{x}$, y reales, $\mathbf{y}$. Estos problemas son de la siguiente forma:

$$Min \quad cx + dy \tag{3.1}$$

$$s.a. \quad Ax + Gy \leq b \tag{3.2}$$

$$x \in \mathbb{Z}^n, y \in \mathbb{R}^m \tag{3.3}$$

donde c, d, A, G son matrices racionales con las dimensiones apropiadas. x e y son las variables del problema. Cada fila (α, γ, β) de (A, G, b) define una restricción. La restricción $(\alpha,$

γ, β) es satisfecha por un punto $(x, y) \in \mathbb{R} \bigtimes \mathbb{R}^n$ si $\alpha x + \gamma y \leq \beta$. Un punto $(x, y) \in \mathbb{Z}^n \bigtimes \mathbb{R}^m$ es *factible* si satisface todas las restricciones del problema. El conjunto de todos los puntos factibles definen la *región factible*. La función a optimizar en 3.1 se denomina *función objetivo* u *objetivo*. Una *solución óptima* es un punto factible con el menor valor objetivo. La *relajación lineal* es una variante del problema original que elimina las restricciones enteras de 3.3, es decir, reemplaza $x \in \mathbb{Z}^n$ por $x \in \mathbb{R}^n$. Dado que los puntos factibles del problema forman un subconjunto de los puntos factibles de su relajación lineal, el óptimo de la relajación provee un límite inferior del valor óptimo del problema. Por tanto, si una solución óptima de la relajación lineal satisface las restricciones enteras, entonces esa solución es también óptima para el problema original. Si la relajación es no factible, entonces el problema es también no factible, y si la relajación es no acotada, entonces el problema original es no acotado o no factible.

El algoritmo *Branch and Bound* sigue una estrategia "divide y vencerás" por la cual la región factible del MIP se divide en subregiones, tras lo que cada subregión es explorada recursivamente. El algoritmo mantiene una lista **L** de subproblemas activos, los problemas de optimización sobre estas subregiones. Sea MIP(k) el subproblema k. El valor objetivo de cualquier solución factible de MIP(k) provee una cota superior del óptimo global. La solución factible con el menor valor objetivo encontrado hasta ahora se denomina *solución incumbente* y su valor objetivo se denota como z^{mejor}. Sea x^k una solución óptima de la relajación lineal de MIP(k) con valor objetivo z^k. Si x^k satisface la restricciones enteras, entonces es una solución óptima de MIP(k) y una solución factible de MIP, y, por tanto, se actualiza z_{mejor} como $\min z^k, z^{mejor}$. En caso contrario, hay dos posibilidades: si $z^k \geq z^{mejor}$, entonces una solución óptima de MIP(k) no puede mejorar z^{mejor}, por tanto el subproblema MIP(k) ya no es tenido en cuenta; por otro lado, si $z^k < z^{mejor}$, entonces MIP(k) debe seguir siendo explorado, lo que se lleva a cabo mediante ramificación (*branching*), i.e., creando $q \geq 2$ nuevo subproblemas MIP($k(i)$), $i = 1, 2, ..., q$ de MIP(k) dividiendo la región factible S^k en q subregiones $S^{k(i)}$, $i = 1, 2, ..., q$. Una forma simple de evitar x^k en los subproblemas de MIP(k) se obtiene seleccionando la variable x_i para la que x_i^k no es entera y creando dos subproblemas; en un subproblema se añade la restricción $x_i \leq \lfloor x_i^k \rfloor$ y en el otro $x_i \geq \lceil x_i^k \rceil$. MIP($k$) es reemplazado por sus subproblemas en la lista **L** de subproblemas activos. El valor más pequeño entre todos los asociados con las relajaciones lineales de los subproblemas activos provee una cota inferior sobre el óptimo global. El algoritmo termina cuando la cota global inferior y la cota global superior son iguales, en cuyo caso la lista **L** de subproblemas queda vacía.

Es conveniente representar el algoritmo *Branch and Bound* como un *árbol de búsqueda* en el que los nodos del árbol representan los subproblemas. El nodo superior del árbol se conoce como *raíz* o *nodo raíz* y representa el MIP original. Los nodos hijo representan los subproblemas obtenidos mediante ramificación. En cualquier momento, los nodos hoja del árbol denotan la lista de subproblemas activos que aún no han sido evaluados.

Dado que el algoritmo *Branch and Bound* termina cuando la cota inferior global y la cota superior global son iguales, los esfuerzos dedicados a hacer el algoritmo más eficiente deben concentrarse en mejorar rápidamente estas cotas globales, i.e., en reducir la cota global superior e incrementar la cota global inferior. Reducir la cota global superior puede lograrse encontrando soluciones mejores durante el proceso de búsqueda. Incrementar la cota global inferior, dado que es el mínimo entre todas las cotas inferiores de los nodos activos, solo

Algorithm 1 Branch and Bound

1: **Inicialización.**
$\mathbf{L} = \{\text{MIP}\}$. $z^{mejor} = \infty$. $x^{mejor} = \emptyset$.
2: **¿Terminación?**
¿Es $\mathbf{L} = \emptyset$. En caso afirmativo, la solución x^{mejor} es óptima.
3: **Selección.**
Elegir y elminar un problema MIP(k) de $\mathbf{L}$.
4: **Evaluación.**
Resolver la relajación lineal LP(k) de MIP(k). Si LP(k) es infactible, ir al paso 1, si no, sea z^k su valor objetivo y x^k su solución.
5: **Poda.**
Si $z^k \geq z^{mejor}$, ir al paso 1. Si x^k no es entero, ir al paso 5, si no sea $z^{mejor} = z^k$, $x^{mejor} = x^k$. Ir al paso 1.
6: **Ramificar (Branch).**
Dividir el conjunto factible S^k de MIP(k) en conjuntos más pequeños $S^{k(i)}$ para $i = 1$, ..., q, tales que $\cup_{i=1}^{q} S^{k(i)} = S^k$ y añadir los subproblemas MIP($k(i)$), $i = 1$, ..., q a $\mathbf{L}$. Ir al paso 1.

se puede alcanzar eligiendo el nodo con la menor cota inferior y mejorando dicha cota del programa lineal. Las cotas de los programas lineales pueden ser mejoradas mediante ramificación, i.e., dividiendo la región factible en piezas más pequeñas, o añadiendo restricciones al subproblema que elminan la solución óptima del programa linal pero mantienen al menos una solución entera óptima (dichas restricciones frecuentemente se denominan *planos de corte*)[1].

3.2. Planos de corte

Como se ha mencionado anteriormente, la clave para resolver MIPs de gran tamaño de forma exitosa en un *framework* basado en *Branch and Bound* es tener unas cotas superior e inferior fuertes sobre el valor objetivo óptimo. Los *solvers* modernos generan planos de corte para reforzar las restricciones de las relajaciones lineales en los nodos del árbol *Branch and Bound*.

Antes de discutir los detalles relacionados con la adición de planos de corte, se hará una revisión rápida del enfoque. Una *inecuación válida* para un MIP es una inecuación que es satisfecha por todas las soluciones factibles. Un *plano de corte*, o simplemente *corte*, es una inecuación válida que no es satisfecha por todos los puntos factibles de la relajación lineal. Si se encuentra un corte violado por una solución del programa lineal, se puede añadir a la formulación para reforzar la relajación. De este modo, se modifica la formulación actual de tal modo que la región factible relajada disminuye en tamaño pero la región factible del MIP no cambia. Entonces la relajación lineal de la nueva formulación se resuelve y el proceso de generación de cortes se repite tantas veces como sea necesario. También es posible añadir planos de corte que eliminen parte de la región factible del MIP, siempre y cuando al menos

[1] La explicación del método Branch & Bound ha sido extraída de (Atamtürk y Savelsbergh, 2005), parafraseada y traducida por el autor.

una solución entera óptima permanezca intacta. Tales cortes son típicamente conocidos como *cortesdeoptimalidad*.

El método branch-and-cut es una generalización del método *Branch and Bound*, en el que se añaden cortes violado a la formulación de los nodos del árbol de búsqueda. Si no se encuentran cortes violados o la efectividad de los planos de cortes en la mejora de la cota del programa lineal se reduce, se ramifica. Los algoritmos branch-and-cut generalizan tanto los algoritmos de planos de corte puros, en los que se añaden cortes hasta que se encuentra una solución entera óptima, i.e., no se ramifica, como los algoritmos *Branch and Bound*, en los que no se añaden cortes.

Naturalmente, un algoritmo branch-and-cut consume mucho tiempo resolviendo relajaciones lineales en los nodos. Sin embargo, el resultado de mejorar las cotas del programa lineal es, generalmente, un árbol de búsqueda significativamente más pequeño. Si los cortes mejoran las cotas lineales significativamente y la adición de los cortes no incrementa los tiempos de obtención de soluciones de los programas lineales demasiado, un algoritmo branch-and-cut cuidadosamente implementado puede ser mucho más rápido que un algoritmo *Branch and Bound*. Encontrar un buen equilibrio entre corte y ramificación es esencial para reducir el tiempo de obtención de soluciones, y esto es muy dependiente del problema [2]

En la figura 3.1 se muestra la relajación lineal del problema:

$$Min \quad -5x_1 - 6x_2 \tag{3.4}$$

$$s.a. \quad x_1 + 2x_2 \leq 7 \tag{3.5}$$

$$2x_1 - x_2 \leq 3 \tag{3.6}$$

$$x_1, x_2 \geq 0, x_1 \in \mathbb{Z}, x_2 \in \mathbb{Z} \tag{3.7}$$

Los puntos en negrita son los puntos factibles del problema entero, mientras que las líneas sólidas delimitan la región factible de la relajación lineal. Para más detalles sobre la resolución paso a paso de este problema se remite al lector a la sección *Integer Programming: Branch and Cut Algorithms* de la segunda edición de (Floudas y Pardalos, 2001).

3.2.1. Planos de corte en el nodo raíz

Se resuelve la relajación lineal del MIP y se identifican las restricciones violadas. Estas restricciones se añaden a la formulación del problema. Se repite este bucle hasta alcanzarse un número máximo de cortes determinado. A continuación se aplica el algoritmo *Branch and Bound* hasta encontrar la solución entera del MIP.

[2] La explicación del método de los planos de corte ha sido extraída de (Atamtürk y Savelsbergh, 2005), parafraseada y traducida por el autor.

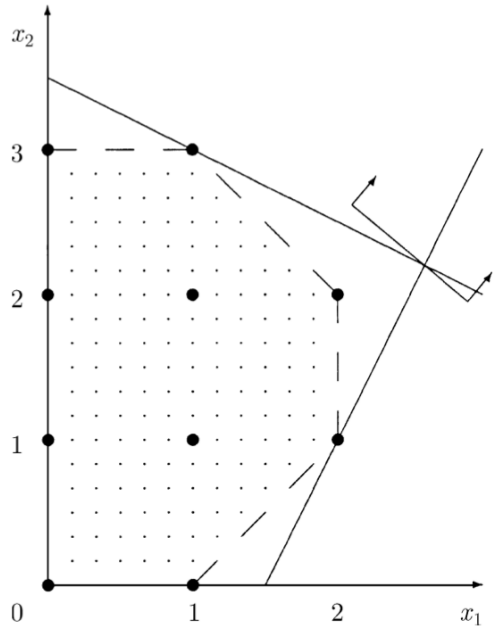

Figura 3.1: Ejemplo del politopo de la región factible resultante de la relajación lineal de un problema entero (Floudas y Pardalos, 2001).

3.2.2. Planos de corte hasta una profundidad dada

Se aplica el bucle de la subsección anterior sobre los subproblemas correspondientes a los nodos hasta una profundidad dada, típicamente 4. A continuación se aplica el procedimiento *Branch and Bound* hasta encontrar la solución entera del MIP.

3.3. Análisis de resultados

La tabla A.1 muestra los resultados del *benchmark* de los métodos exactos estudiados en este trabajo. La figura 3.2 resume el *índice de optimalidad* para los métodos exactos, en la que se aprecia que la formulación fuerte proporciona los mejores resultados de los 4 métodos estudiados, con un índice de 0.80. El resto de métodos ofrecen un rendimiento similar, con un 0.72 para la formulación débil y un 0.76 para planos de corte y *Branch and Cut*. La formulación débil resulta ser el peor método: aunque tiene como beneficio que el problema es más ligero que en la formulación fuerte, i.e. el número de restricciones es mucho menor, por lo que la fase de relajación lineal toma mucho menos tiempo. Véase, por ejemplo, el problema MT1, que tiene 4000 restricciones con la formulación débil, frente a las 4002000 que tiene con la formulación fuerte. Este sobrecoste en la fase de relajación lineal permite una mejor acotación del óptimo entero, por lo que la fase de ramificación será mucho más rápida que

con la formulación débil, que tarda más en cerrar el hueco. Esto se traduce, dentro de la metodología aplicada en este trabajo, en que la formulación fuerte es capaz de encontrar el óptimo en más ficheros en el tiempo proporcionado para la fase *Branch and Bound* que la débil. El método de planos de corte empieza con la formulación débil y añade restricciones según los cortes son violados, de modo que la región factible está más restringida que la de la formulación débil pero menos que la fuerte, lo que conlleva algunas mejoras con respecto a esta primera. No hay ninguna mejora a nivel de *índice de optimalidad* con el método *Branch and Cut* con respecto al de planos de corte.

En la figura 3.3 se analiza el rendimiento de cada método por tipo de fichero. De este modo se puede afirmar que para los ficheros más sencillos, cap y UnifS, no existen diferencias entre los métodos, mientras que para los ficheros difíciles, M*, se puede apreciar que la formulación fuerte alcanza resultados mucho mejores que sus competidores. Dado que estas instancias se asemejan a los problemas reales, para este problema debería optarse por la fomrulación fuerte si se desea resolver el problema con un *solver* comercial. El rendimiento, sin embargo, no es demasiado bueno, pues tan solo alcanza el óptimo para el 36 % de los problemas, entre los que no se encuentran los problemas más grandes, que son los de más interés. El resto de métodos ni siquiera deberían ser tenidos en cuenta, a tenor de los resultados: la formulación débil no alcanza el óptimo en ninguno de los ficheros M* y planos de corte y *Branch and Cut* tan solo resuelven el 18 %.

La figura 3.4, por otro lado, ilustra el tiempo de finalización de los ficheros para los que se alcanzaron el óptimo con los distintos métodos exactos. En estos términos, a priori, el peor método es la formulación fuerte, que es unas 4 veces más lenta que el método de los planos de corte y el algoritmo *Branch and Cut*. Sin embargo, teniendo en cuenta que no todos los métodos alcanzan el óptimo en los mismos ficheros, hay que realizar un análisis más minucioso. Así, por ejemplo, se puede ver que la formulación débil es más lenta para los ficheros UnifS que la fuerte y toma un tiempo similar para los ficheros cap. Esto hace muy desaconsejable la formulación débil, pues para este problema es lenta y tiene un índice de optimalidad bajo. Los métodos de planos de corte y *Branch and Cut* presentan una distribución de tiempos similar con respecto a la formulación fuerte en los ficheros cap y UnifS, por lo que las diferencias se encuentran en las instancias M*. Con el apoyo de los datos originales de la tabla A.1 se ve que las nubes de puntos de los ficheros M* de planos de corte y *Branch and Cut* se corresponden con instancias MO*, al igual que la nube de puntos que se entremezcla con los ficheros UnifS para la formulación fuerte, i.e se resuelven en tiempos similares con los tres métodos. Sin embargo, hay otra nube de puntos de ficheros M* en la formulación fuerte con tiempos mayores. Estos corresponden a ficheros MP*, para los que el resto de métodos no alcanzaron el óptimo.

Por tanto, teniendo en cuenta los resultados de índice de optimalidad y tiempos de los cuatro métodos estudiados se aconseja la formulación fuerte cuando se trate de resolver instancias del problema UFLP utilizando un *solver* de programación entera. Se desaconseja encarecidamente la formulación débil y no parece existir ninguna razón para utilizar los métodos de planos de cortes y *Branch and Cut* para este caso, a menos que se desee resolver problemas pequeños y exista algún motivo de peso para no usar la formulación fuerte.

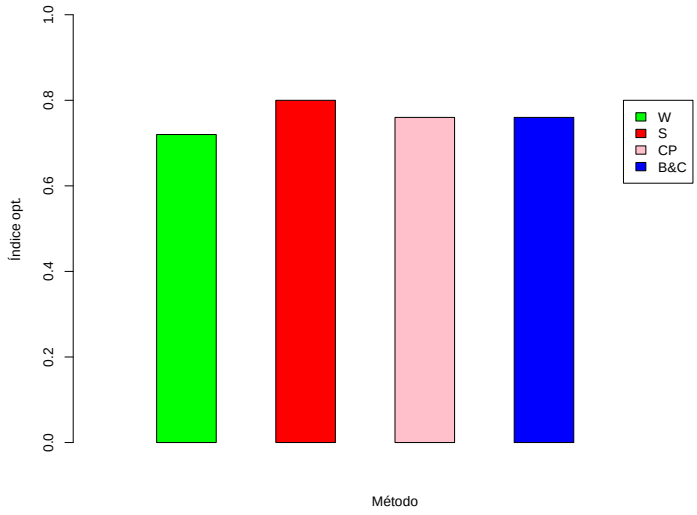

Figura 3.2: Índice de optimalidad de cada método exacto.

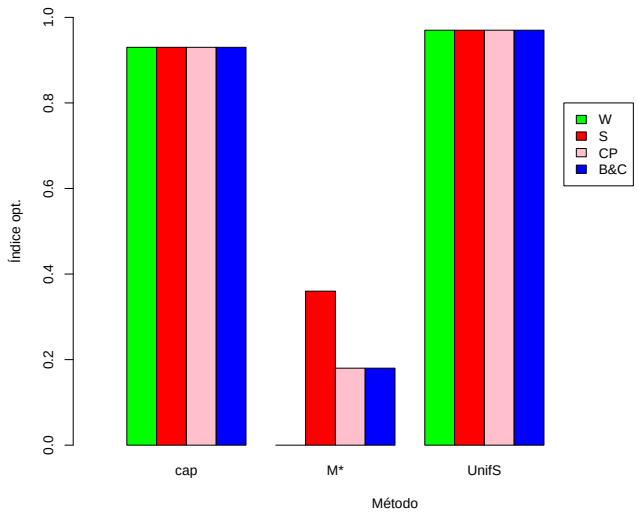

Figura 3.3: Índice de optimalidad de cada método exacto por tipo de fichero.

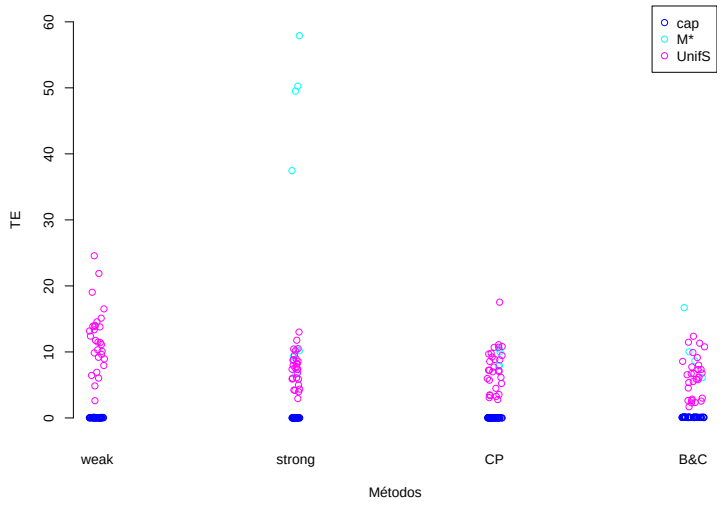

Figura 3.4: Tiempo de finalización de cada método exacto por tipo de fichero.

3.4. Función objetivo

La función objetivo del problema (2.1 ó 2.6) es una función cóncava. En la figura 2.1 se muestra una estimación de la forma de la función, según Daskin (Daskin, 2011). Tal y como se muestra en ella, cuando solo se abre una instalación, los costos fijos son mínimos (pues se ha solo existe el costo fijo de esta única instalación) y los costos de transporte son máximos, pues toda la demanda es satisfecha por la misma instalación, y esto engloba la de aquéllos clientes para los que el costo es bajo y para los que es alto. En el otro extremo está la situación en la que se abren todas las instalaciones. Los costos fijos son máximos, mientras que los costos de transporte son mínimos, pues a cada cliente se le puede servir con la instalación que le de un menor coste. Entre estos dos escenarios se encuentra el óptimo del problema y, es en base a la forma de esta función, cómo se desarrollan las heurísticas de construcción del capítulo 4.

Capítulo 4

Heurísticas de construcción

4.1. ¿Qué es una heurística de construcción?

Una heurística de construcción es un algoritmo que llega a una solución factible siguiendo algún criterio heurístico, esto es, de aproximación con conocimiento limitado, y que sirve de base para otro algoritmo que tratará de mejorar esta solución con objeto de llegar al valor objetivo óptimo. En este trabajo servirán de base para las metaheurísticas, explicadas en el capítulo 6.

Se han estudiado dos heurísticas de construcción para la resolución del problema UFLP: ADD y DROP. Ambas son algoritmos greedy. Los algoritmos greedy, o voraces, se caracterizan por elegir la mejor solución actual y no retroceder para evaluar posibles soluciones pasadas. Esto puede conducir a mínimos locales, lo que impide que alcancen siempre el óptimo de la función objetivo. Esto puede hacerles inadecuados para algunas aplicaciones. Sin embargo, tienen como ventaja sobre los métodos exactos su baja carga computacional, lo que permite obtener una solución rápidamente. En este trabajo esta solución sirve como punto de partida para las metaheurísticas que se describen en el siguiente capítulo, que mejoran el valor objetivo de esta solución y, en muchos casos, llegan al óptimo.

4.2. Heurística ADD

En la heurística ADD se pretende alcanzar el mínimo de la función objetivo por la izquierda. Para ello se comienza con 0 instalaciones y se elige la instalación que minimiza la función objetivo para su apertura. Se continúa abriendo instalaciones mientras la función objetivo decrezca, pues, tal y como ilustra la gráfica (figura 2.1), aún no se habrá alcanzado el mínimo. Si al abrir una instalación la función objetivo crece, se ha sobrepasado el mínimo, por lo que no se abre dicha instalación y el algoritmo termina.

Algorithm 2 Heurística ADD

```
1: function addGreedy
2:     Sea z_mejor = ∞ el mejor valor objetivo encontrado hasta el momento
3:     Sea V[1...n] un nuevo array auxiliar
4:     Sea S[1...n] un nuevo array binario solución
5:     Sea S_aux[1...n] una copia de S
6:     for j = 1 to n do
7:         if S[j] = 0 then
8:             S_aux[j] = 1                                   ▷ Se abre j
9:             V[j] = valorObjetivo(S_aux)
10:            S_aux[j] = 0                ▷ S_aux vuelve a ser una copia exacta de S
11:        else
12:            V[j] = ∞
13:        end if
14:    end for
15:    k = argmin(V)                ▷ Instalación que mejora más el valor objetivo
16:    if V[k] > z_mejor then
17:        return
18:    else
19:        z_mejor =V[k]
20:        S[k] = 1                                           ▷ Se abre k
21:    end if
22: end function
```

4.3. Heurística DROP

Al contrario que en la heurística ADD, aquí se pretende alcanzar el mínimo de la función objetivo por la derecha. Para ello se abren las n instalaciones y se asigna a cada cliente aquella que le proporcione servicio a menor coste c_{ij}. A continuación se comienza a cerrar instalaciones hasta que la función objetivo comience a crecer. En este punto se habrá sobrepasado el mínimo de la función, por lo que, al igual que en la heurística ADD, no se abre la instalación y el algoritmo termina.

4.4. Benchmark

Los valores objetivo de las soluciones y el tiempo de ejecución de los algoritmos se muestran en la tabla 4.1. Se señala, además, en qué ficheros el valor objetivo fue más pequeño para ADD con el símbolo + y con - en qué casos DROP fue mejor. Para aquellos ficheros en los que ambos algoritmos llegaron al mismo valor objetivo se señala con =. En el 22.83 % de los ficheros la heurística ADD obtuvo mejores resultados, mientras que DROP llega a soluciones mejores en el 73.91 % de los ficheros estudiados. En el resto ambos algoritmos llegaron al mismo valor objetivo.

Algorithm 3 Heurística DROP

```
1: function dropGreedy
2:     Sea z_mejor = ∞ el mejor valor objetivo encontrado hasta el momento
3:     Sea V[1...n] un nuevo array auxiliar
4:     Sea S[1...n] un nuevo array binario solución
5:     for j = 1 to n do
6:         S[j] = 1                                  ▷ Se abren todas las instalaciones
7:     end for
8:     Sea S_aux[1...n] una copia de S
9:     for j = 1 to n do
10:        if S[j] = 1 then
11:            S_aux[j] = 0                          ▷ Se cierra j
12:            V[j] = valorObjetivo(S_aux)
13:            S_aux[j] = 1                          ▷ S_aux vuelve a ser una copia exacta de S
14:        else
15:            V[j] = ∞
16:        end if
17:    end for
18:    k = argmin(V)                                 ▷ Instalación que mejora más el valor objetivo
19:    if V[k] > z_mejor then
20:        return
21:    else
22:        z_mejor =V[k]
23:        S[k] = 0                                  ▷ Se cierra k
24:    end if
25: end function
```

Cuadro 4.1: Resultados benchmark heurísticas ADD y DROP

Datos	**z(ADD)**	**z(DROP)**	**t(ADD)**	**t(DROP)**
-cap101.txt	797 508.73	796 648.44	0.0003	0.0003
-cap102.txt	855 971.75	855 466.85	0.0002	0.0003
-cap103.txt	895 027.19	893 782.11	0.0001	0.0003
+cap104.txt	928 941.75	934 586.98	0.0001	0.0005
-cap111.txt	794 299.85	793 439.56	0.0007	0.0013
-cap112.txt	852 762.88	852 257.98	0.0005	0.0018
-cap113.txt	894 095.76	893 782.11	0.0003	0.0018
+cap114.txt	928 941.75	934 586.98	0.0002	0.0017
-cap121.txt	794 299.85	793 439.56	0.0005	0.0012
-cap122.txt	852 762.88	852 257.98	0.0004	0.0013
-cap123.txt	894 095.76	893 782.11	0.0003	0.0013
+cap124.txt	928 941.75	934 586.98	0.0002	0.0019
-cap131.txt	794 299.85	793 439.56	0.0005	0.0014
-cap132.txt	852 762.88	852 257.98	0.0005	0.0013
-cap133.txt	894 095.76	893 782.11	0.0003	0.0014
+cap134.txt	928 941.75	934 586.98	0.0002	0.0014
=cap41.txt	932 615.75	932 615.75	0.0001	0.0001
-cap42.txt	981 538.85	977 799.40	0.0001	0.0001
-cap43.txt	1 012 476.98	1 010 641.45	0.0001	0.0001
+cap44.txt	1 034 976.98	1 037 717.08	0.0000	0.0002
-cap51.txt	1 012 476.98	1 010 641.45	0.0000	0.0001
=cap61.txt	932 615.75	932 615.75	0.0001	0.0001
-cap62.txt	981 538.85	977 799.40	0.0001	0.0001
-cap63.txt	1 012 476.98	1 010 641.45	0.0001	0.0001
+cap64.txt	1 034 976.98	1 037 717.08	0.0001	0.0002
=cap71.txt	932 615.75	932 615.75	0.0001	0.0001
-cap72.txt	981 538.85	977 799.40	0.0001	0.0001
-cap73.txt	1 012 476.98	1 010 641.45	0.0001	0.0002
+cap74.txt	1 034 976.98	1 037 717.08	0.0001	0.0002
-cap81.txt	797 508.73	796 648.44	0.0002	0.0003
-cap82.txt	855 971.75	855 466.85	0.0002	0.0003
-cap83.txt	895 027.19	893 782.11	0.0001	0.0004
+cap84.txt	928 941.75	934 586.98	0.0001	0.0004
-cap91.txt	797 508.73	796 648.44	0.0002	0.0002
-cap92.txt	855 971.75	855 466.85	0.0001	0.0003
-cap93.txt	895 027.19	893 782.11	0.0001	0.0004
+cap94.txt	928 941.75	934 586.98	0.0001	0.0004
+capa.txt	17 902 353.24	18 348 992.54	0.0112	0.0906
-capb.txt	13 131 893.84	13 114 399.65	0.0129	0.0877
-capc.txt	11 947 717.76	11 647 844.40	0.0129	0.0700
-MO1	1479.26	1305.95	0.0009	0.0102
-MO2	1587.79	1432.36	0.0007	0.0120
-MO3	1674.45	1521.47	0.0007	0.0138

-MO4	1617.50	1442.24	0.0012	0.0128
-MO5	1592.34	1411.11	0.0009	0.0110
-MP1	3119.16	2695.72	0.0030	0.0793
-MP2	3238.66	2904.86	0.0040	0.0932
-MP3	2963.92	2623.71	0.0032	0.0834
-MP4	3361.71	2945.65	0.0030	0.0879
-MP5	3250.79	2939.95	0.0033	0.0848
-MQ1	4788.63	4091.01	0.0083	0.2675
-MQ2	4681.45	4028.33	0.0107	0.2621
-MQ3	4358.08	4275.43	0.0113	0.2566
-MQ4	4239.23	4235.15	0.0119	0.3403
-MQ5	4147.69	4103.75	0.0082	0.2616
-MR1	2973.83	2609.13	0.0240	1.2554
-MR2	3087.79	2686.83	0.0249	1.2411
-MR3	3126.40	2794.41	0.0279	1.2262
-MR4	3160.91	2756.04	0.0301	1.2368
-MR5	2916.15	2510.14	0.0309	1.2337
-MS1	5297.52	5283.76	0.1915	12.1825
-MT1	10 127.25	10 108.34	0.8437	107.1687
-1023UnifS.txt	76 292.00	75 710.00	0.0027	0.0098
-1123UnifS.txt	81 816.00	80 207.00	0.0023	0.0116
-1223UnifS.txt	75 103.00	74 893.00	0.0019	0.0097
+123UnifS.txt	74 621.00	77 960.00	0.0019	0.0121
+1323UnifS.txt	75 992.00	79 729.00	0.0018	0.0096
+1423UnifS.txt	76 299.00	79 437.00	0.0022	0.0096
+1523UnifS.txt	79 971.00	81 946.00	0.0023	0.0108
-1623UnifS.txt	78 084.00	77 855.00	0.0020	0.0106
-1723UnifS.txt	79 482.00	78 364.00	0.0022	0.0112
+1823UnifS.txt	76 683.00	76 925.00	0.0020	0.0097
+1923UnifS.txt	76 980.00	77 036.00	0.0026	0.0101
-2023UnifS.txt	80 825.00	77 709.00	0.0029	0.0100
+2123UnifS.txt	77 190.00	77 350.00	0.0021	0.0096
-2223UnifS.txt	78 558.00	75 861.00	0.0020	0.0127
+223UnifS.txt	75 977.00	77 470.00	0.0022	0.0129
-2323UnifS.txt	76 506.00	75 693.00	0.0026	0.0119
-2423UnifS.txt	81 316.00	78 916.00	0.0021	0.0114
+2523UnifS.txt	78 238.00	78 788.00	0.0028	0.0097
-2623UnifS.txt	76 449.00	76 132.00	0.0022	0.0107
-2723UnifS.txt	79 783.00	79 711.00	0.0027	0.0099
-2823UnifS.txt	79 648.00	76 839.00	0.0029	0.0096
-2923UnifS.txt	80 636.00	77 203.00	0.0023	0.0113
+3023UnifS.txt	77 062.00	78 206.00	0.0026	0.0120
-323UnifS.txt	80 482.00	74 305.00	0.0028	0.0119
-423UnifS.txt	79 414.00	78 562.00	0.0020	0.0115
-523UnifS.txt	76 876.00	76 619.00	0.0022	0.0128
+623UnifS.txt	76 008.00	76 077.00	0.0022	0.0099
-723UnifS.txt	76 980.00	74 733.00	0.0021	0.0097

-823UnifS.txt	74 813.00	73 408.00	0.0023	0.0117
-923UnifS.txt	79 009.00	77 528.00	0.0027	0.0095

Capítulo 5

Búsqueda local

5.1. Concepto de búsqueda local

La búsqueda local es una heurística de mejora utilizada en problemas de optimización para la resolución de problemas computacionalmente complejos. Se basa en la realización de movimientos a partir de una solución inicial, que en este trabajo proviene siempre de una heurística de construcción greedy (ADD o DROP), manteniéndose siempre dentro del espacio de soluciones, tal y como se describe en el cuadro algorítmico 4 (Brito, Pérez y González, 2004). Nótese que aquí se describe la búsqueda local *Best* i.e. se busca la mejor solución vecina, pero existen otros enfoques, como puede ser *First*, que trabaja con la primera solución vecina encontrada, o *Worst*, que escoge la solución que mejore menos la solución actual.

Algorithm 4 Búsqueda local

Inicialización. Seleccionar la estructura de entorno $\mathcal{N}$ a utilizar en la búsqueda y encontrar una solución inical x

Repetir los siguientes pasos hasta verificarse el criterio de parada (por ejemplo, encontrar un óptimo local):

a Encontrar la mejor solución vecina x' de x ($x' \in \mathcal{N}$).

b Si x' no es mejor que x, parar. En otro caso, hacer $x = x'$ y volver al paso (a).

5.2. Estructura de entorno

Dado el problema (S, f), (donde S es el espacio de soluciones y f la función objetivo) una estructura de entorno es una función $\mathcal{N} : S \to 2^S = \{X \mid X \subseteq S\}$ que asocia a cada solución $x \in S$ un conjunto $\mathcal{N}(x) \subset S$ de soluciones cercanas a x. El conjunto $\mathcal{N}(x)$ se denomina *entorno* de x, mientras que cada $y \in \mathcal{N}(x)$ será una *solución vecina* de x (Brito, Pérez y González, 2004).

Esta estructura de entorno es determinada por el investigador, y de ella dependerá la calidad de las soluciones encontradas. Algunos ejemplos de estructuras de entorno ampliamente conocidas son:

- k-intercambios: consiste en el intercambio del estado de k elementos del problema (e.g. en el problema de la mochila, intercambiar k elementos que se encuentran en la mochila por k que no lo están). La versión de 2-intercambios es la más común.
- k-opt: es una estructura de entornos diseñada específicamente para el problema TSP (*Travelling Salesman Problem*) o *Problema del viajante*. Es una versión más general de la estructura de k-intercambios, basado en eliminar k-arcos con $k > 1$ y combinar las k posibles uniones de la mejor forma posible para reducir la distancia recorrida.

Como ejemplo formal de estructura de entorno tómese el problema de la mochila y los k-intercambios. Sea una solución $s = (s_1, s_2, \ldots, s_i \ldots, s_j \ldots, s_n)$:

$$\mathcal{N}(s) = \{s' = (s'_1, s'_2, \ldots, s'_i, \ldots, s'_j \ldots, s'_n) \mid i \neq j, s_i = s'_j = 0, s_j = s'_i = 1\} \tag{5.1}$$

Esto es, son vecinas de la solución s todas aquellas s' para las cuales el elemento i se encuentra en s pero no en s' y el elemento j no se encuentra en s pero sí en s', lo que posibilita el intercambio entre i, que sale de la mochila, y j, que entra en la mochila.

5.3. VNS - Variable Neighbourhood Search

La VNS (*Variable Neighbourhood Search*) o búsqueda en entornos variables es una técnica de búsqueda local que consiste en buscar soluciones en el entorno de una solución inicial s variando la estructura de entorno $\mathcal{N}x$. De este modo se puede escapar de óptimos locales, dado que una solución puede ser un óptimo local para un entorno pero no necesariamente para otro. El pseudocódigo de VNS se muestra en el cuadro algorítmico 5 (Brito, Pérez y González, 2004).

Algorithm 5 VNS

Inicialización. Seleccionar el conjunto de estructuras de entorno $\mathcal{N}_k, k = 1, \ldots, k_{max}$ que se usarán en la búsqueda y encontrar una solución inicial x. Elegir, también, el criterio de parada a emplear.

Repetir hasta que no se obtenga mejora, la siguiente secuencia

1. Hacer $k \leftarrow 1$
2. Repetir, hasta que $k = k_{max}$, los pasos:
   a *Exploración del entorno.* Encontrar la mejor solución x' del k-ésimo entorno de x.
   b *Moverse o no*: si la solución obtenida x' es mejor que x, hacer $x \leftarrow x'$ y $k \leftarrow 1$; en otro caso, hacer $k \leftarrow k + 1$.

Esta es la forma más básica de aplicación del método VNS, denominado Descenso por Entornos Variables, en el que la selección de los entornos se hace de forma determinista. El uso de varios entornos permite obtener una solución más cercana al mínimo global que la encontrada con la búsqueda local, dado que se trata del mínimo local repecto de las k'_{max} estructuras de entorno consideradas.

A partir del método básico de VNS se han desarrollado múltiples variantes, que a menudo incorporan la aleatorización, como en el caso de la Búsqueda por Entornos Variables reducida, cuyo pseudocódigo se muestra en el cuadro algorítmico 6 (Brito, Pérez y González, 2004).

Algorithm 6 VNS

Inicialización. Seleccionar el conjunto de estructuras de entorno $\mathcal{N}_k, k = 1, \ldots, k_{max}$ que se usarán en la búsquea y encontrar una solucón inicial x. Elegir, también, el criterio de parada a emplear.

Repetir los siguientes pasos hasta verificarse el criterio de parada:

a *Agitación*: generar al azar una solución x' del k-ésimo entorno de x ($x' \in \mathcal{N}_k(x)$).

b *Moverse o no*: si la solución x'' mejora la mejor obtenida, hacer $x = x''$ y continuar la búsqueda con $\mathcal{N}_1$ ($k = 1$). En otro caso, hacer $k = k + 1$.

5.4. Utilización de VNS en este trabajo

El VNS implementado en este trabajo proporciona una solución vecina según el entorno seleccionado (lo que en el algoritmo de la Búsqueda por Entornos Variables Reducida equivaldría a una sola iteración) y es la metaheurística la que realiza o no el movimiento, según su mecanismo de aceptación de soluciones candidatas, e.g. la probabilidad que depende de la temperatura en Simulated Annealing. De este modo se pretende reducir la carga computacional que supone aplicar un método de búsqueda local completo que podría ralentizar en exceso las metaheurísticas para los problemas más grandes, MR*, MS* y MT*. Estos no aparecen en los artículos revisados para la elaboración de este trabajo, lo que permite el uso de una búsqueda local más profunda. En cualquier caso, este enfoque parece funcionar, a tenor de los resultados obtenidos (véase el capítulo 6), que son razonables en calidad y tiempo, incluso para los problemas de mayor dimensión.

La estructura de entornos escogida es la descrita en el artículo de Yigit et al. (Yigit, Aydin y Turkbey, 2004), que introduce la aleatorización en la selección del entorno. Esta depende de una probabilidad ρ y el número de instalaciones abiertas $\lambda(S)$.

$$
\begin{aligned}
&intercambio() && (\lambda(S) = 1 \cap 0 \leq \rho < 0{,}7) \cup (\lambda(S) > 1 \cap 0 \leq \rho \leq 0{,}5) \\
&abrir() && (\lambda(S) = 1 \cap 0{,}7 \leq \rho < 1) \cup (\lambda(S) > 1 \cap 0{,}5 \leq \rho \leq 0{,}7) \\
&cerrar() && (\lambda(S) = n) \cup (\lambda(S) < n \cup 0{,}7 \leq \rho \leq 1)
\end{aligned}
$$

En relación con el esquema propuesto por Brito et al. del cuadro algorítmico 6, se utilizan

$k_{max} = 3$ estructuras de entorno. La selección del entorno no depende de un orden preestablecido de los mismos, como se refleja en el cuadro, sino que depende de ρ y del número de instalaciones abiertas en la solución actual.

Los nombres de las estructuras de entorno son autoexplicativos, aunque se describirán de forma somera para evitar confusiones acerca del número de instalaciones sobre el que operan. Así, el entorno *intercambio()* (conocido en la literatura por el término inglés *swap*) cierra una instalación que se encuentre abierta en la instalación actual y abre otra que se encuentre cerrada, *abrir()* (*add*) abre una instalación que se encuentre actualmente cerrada, mientras que *cerrar()* (*drop*) cierra una instalación que se encuentre actualmente abierta.

Nótese que cuando solo hay una instalación abierta no se permite cerrar ninguna, mientras que cuando todas están abiertas se fuerza a cerrar alguna. De este modo nunca se llega a la situación en la que no haya ninguna instalación abierta, lo que llevaría a un coste virtualmente infinito, pues no se está sirviendo la demanda, y que, desde el punto de vista computacional, podría llevar a excepciones en la ejecución del algoritmo que llevarían al tratamiento de excepciones. Del mismo modo no se permite la situación, siempre subóptima de abrir todas las instalaciones. Esto sería muy favorable para los clientes, pues se les serviría al menor coste posible, pero sería fatídico para la empresa que dispone las instalaciones, pues se maximizaría la suma de costos fijos.

El método VNS implementado para este trabajo ha sido utilizado como técnica de búsqueda local en las metaheurísticas Simulated Annealing, Threshold Accepting, Late Acceptance, Old Bachelor y Demon Algorithm. El resto de metaheurísticas, Tabu Search y Algoritmo Genético, tienen sus propias técnicas de búsqueda local, que se describen en las secciones correspondientes del capítulo 6.

Capítulo 6

Metaheurísticas

6.1. Simulated Annealing

Simulated Annealing o Recocido Simulado es una metaheurística inspirada en la metalurgia y la termodinámica: cuando el hierro fundido es enfriado muy despacio tiende a solidificar en una estructura de energía mínima. De esta manera, en las primeras iteraciones del algoritmo, a alta temperatura, se admite casi cualquier solución, aunque esta empeore la función objetivo, de modo que se pueda explorar una región grande del espacio de soluciones. De este modo aumenta la probabilidad de encontrar el óptimo de la función objetivo. A medida que el algoritmo avanza, la calidad de las soluciones exigida aumenta. Esto se controla por medio de una función de probabilidad que varía con la temperatura, que disminuye a medida que se ejecutan iteraciones. Las soluciones que mejoran el valor objetivo actual siempre se admiten, mientras que la adopción de una solución que empeora la función objetivo viene determinada por esta probabilidad.

Simulated Annealing es un algoritmo que proporciona soluciones muy cercanas al óptimo en gran variedad de problemas, entre los que se encuentra el que es objeto de estudio de este trabajo. Es preciso describir esta metaheurística en primer lugar, dado que parte de las tratadas en este trabajo derivan de esta.

Aunque Simulated Annealing permite obtener soluciones en un tiempo menor que el que requieren los métodos exactos que, para problemas grandes, pueden ser impracticables, el coste computacional es alto, debido, principalmente, al cálculo de la probabilidad de admisión. Por ese motivo surgen otras metaheurísticas que tratan de solventar este problema, como pueden ser Threshold Accepting y Demon Algorithm.

El pseudocódigo de Simulated Annealing se muestra en el cuadro algorítmico 7. La aceptación de una solución que empeora el valor objetivo actual viene determinada por *probAnnealing()*. Típicamente, esta función retornará un valor aleatorio de la distribución de Boltzmann.

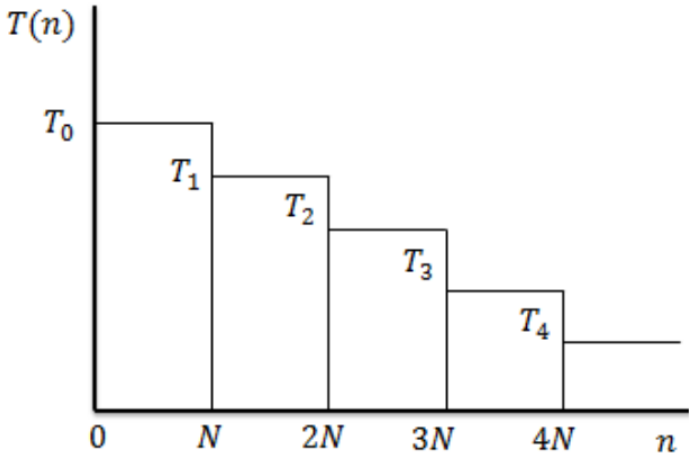

Figura 6.1: Política de enfriamiento aplicada en Simulated Annealing (Pirlot, 1996)

Existen varias formas de actualizar la temperatura. En general existen dos enfoques: decrecimiento homogéneo, que reduce la temperatura por escenarios (se trata de alcanzar el equilibrio en el nivel de temperatura actual antes de enfriar de nuevo) y el heterogéneo, que disminuye la temperatura en cada iteración (de acuerdo con una ley geométrica, por ejemplo) (Deroussi, 2016). En este trabajo se ha aplicado una mezcla de ambos. Se mantiene la temperatura durante N iteraciones, y esta disminuye de acuerdo con una ley geométrica de parámetro α. Esta política constituye un decrecimiento escalonado de la temperatura, tal y como se refleja en la figura 6.1.

6.1.1. Benchmark

En la tabla 6.2 se muestran los resultados del *benchmark* de Simulated Annealing con solución base ADD y en la tabla 6.3 se presentan los resultados para el benchmark con solución base DROP. Por claridad, se ha referenciado cada conjunto de parámetros con un índice que se muestra en la tabla 6.1.

Partiendo de la solución de la heurística ADD (tabla 6.2) el 98.91 % de los ficheros alcanzaron el óptimo o mejoraron la mejor solución encontrada hasta la fecha, mientras que el 1.09 % restante no llegaron al óptimo. Tomando como solución base la proporcionada por la heurística DROP (tabla 6.3) se alcanzó o mejoró el óptimo actual en el 97.83 % de los ficheros y no se llegó al óptimo en el 2.17 % restante.

En la figura 6.2 se puede ver no hay grandes diferencias entre las soluciones que parten de la heurística ADD y las que parten de la heurística DROP, pues la mayoría de puntos para ambas situaciones coinciden con un tiempo de solución cercano a 0. Esto es así porque las heurísticas son capaces de encontrar el óptimo o un valor objetivo muy próximo a él, por lo

Algorithm 7 Simulated Annealing

```
1: function SIMULATEDANNEALING
2:     Sea z_mejor = ∞ el mejor valor objetivo encontrado hasta el momento
3:     Sea z_n el valor objetivo de la iteración actual
4:     Sea z_cand el valor objetivo de la solución candidata de la búsqueda local
5:     Sea S_n la solución de la iteración actual
6:     Sea S_cand la solución candidata de la búsqueda local
7:     Sea S_mejor la mejor solución encontrada
8:     Se inicializa S_mejor y z_mejor
9:     Se copia S_mejor a S_n
10:    while fin = 0 do          ▷ Iterar hasta que se cumpla la condición de terminación
11:        busquedaLocal(S_n)                          ▷ S_cand guarda el movimiento
12:        z_cand = valorObjetivo(S_cand)
13:        if z_cand ≤ z_mejor then
14:            S_n = S_cand
15:            z_n = z_cand
16:            if z_n < z_mejor then
17:                S_mejor = S_n
18:                z_mejor = z_n
19:            end if
20:        else
21:            Sea p un valor aleatorio en [0,1]
22:            if p ≤ probAnnealing() then
23:                S_n = S_cand
24:                z_n = z_cand
25:            end if
26:        end if
27:        actualizarTemperatura()          ▷ La actualización depende de una política
28:    end while
29: end function
```

Cuadro 6.1: Parámetros benchmark Simulated Annealing

Índice	T	α	K	N
0	10000	0.95	800	800
1	10000	0.95	800	1000
2	10000	0.95	800	1500
3	10000	0.8	800	800
4	10000	0.8	800	1000
5	10000	0.8	800	1500
6	10000	0.9	800	800
7	10000	0.9	800	1000
8	600	0.95	800	800
9	1500	0.95	800	800
10	6000	0.95	800	800
11	20000	0.95	800	800
12	1500	0.95	800	800
13	1500	0.95	1500	800
14	1500	0.95	4000	800

que la metaheurística puede llegar al óptimo en un tiempo despreciable. Las diferencias más importantes se aprecian en los ficheros M*. Así, se puede apreciar en el gráfico como para los ficheros MQ1-4 la heurística DROP encuentra el óptimo, por lo que el tiempo de solución de la metaheurística es 0. En cambio, la heurística ADD no llega al óptimo, por lo que es necesario correr la metaheurística por un tiempo superior a 1 segundo. Este tiempo, en general, no será lo suficientemente significativo para decantarse por una de las dos heurísticas. Este patrón aparecerá en todos los gráficos, dado que en todos los casos a partir de la solución DROP la metaheurística no realiza ninguna mejora.Para el fichero MR4 el tiempo de solución con base ADD es de más de 4 segundos, mientras que la heurística DROP ya encuentra el óptimo. En cambio, para el fichero MR5 el tiempo de solución a partir de DROP es apróximadamente el doble que a partir de ADD, más de 6 segundos. Salvo este último caso, para la mayoría de ficheros, Simulated Annealing llega antes al valor objetivo óptimo con la heurística DROP.

Simulated Annealing alcanzó el óptimo o mejoró el óptimo teórico actual en el 97.82 % de los ficheros a partir de la solución de ADD y en el 98.91 % a partir de la solución de DROP.

Cuadro 6.2: Resultados benchmark Simulated Annealing - solución base ADD

Datos	BST	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	BST(TS)	BST(TE)
*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap113	893 076.71	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap121	793 439.56	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap123	893 076.71	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.01	0.01
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap133	893 076.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02

*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.04
*capb	12 979 071.58	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	0.03	0.06
#capc	11 509 361.66	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.11	0.16
!MO1	1254.44	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.56	0.66
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.25	0.25
!MO3	1494.40	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.21	0.25
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.23	0.24
*MO5	1408.77	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.23	0.25
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.67	0.74
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.68	0.73
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.73	0.76
*MP4	2938.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.71	0.73
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.70	0.75
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.36	1.45
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.31	1.37
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.41	1.46
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.37	1.45
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.35	1.40
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.38	3.54
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.43	3.52
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.40	3.51
*MR4	2756.04	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	4.20	4.38
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.27	3.47
*MS1	5283.76	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	11.21	11.88
!MT1	10 069.80	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	✗	✓	74.59	88.56
*1023UnifS	71 840.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.28	0.28
*1123UnifS	75 063.00	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	0.03	0.05
*1223UnifS	71 756.00	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	0.04	0.06
*123UnifS	71 342.00	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	0.09	0.12
*1323UnifS	72 414.00	✗	✗	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✗	✓	✓	0.05	0.08
*1423UnifS	72 632.00	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	0.09	0.09
!1523UnifS	77 566.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	0.11	0.12
*1623UnifS	74 344.00	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗	✗	0.10	0.12
*1723UnifS	73 847.00	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	0.05	0.06
*1823UnifS	71 903.00	✗	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.05	0.07
*1923UnifS	75 520.00	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	0.03	0.06
*2023UnifS	72 458.00	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✓	✓	0.06	0.06
*2123UnifS	72 631.00	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	0.04	0.05
*2223UnifS	71 945.00	✗	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	0.07	0.11
*223UnifS	74 148.00	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✓	✗	✓	✓	0.03	0.07
*2323UnifS	71 335.00	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓	✓	✗	✗	✓	0.04	0.06
*2423UnifS	75 342.00	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	0.13	0.15
*2523UnifS	73 807.00	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✗	0.11	0.16
*2623UnifS	70 929.00	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	0.05	0.06
*2723UnifS	77 314.00	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.04	0.09
*2823UnifS	75 025.00	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓	✓	0.11	0.15
*2923UnifS	75 279.00	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓	✗	0.05	0.08
*3023UnifS	75 654.00	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.41	0.46
*323UnifS	73 319.00	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓	0.05	0.08
*423UnifS	74 979.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	0.09	0.12
*523UnifS	69 251.00	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.01	0.06
*623UnifS	72 992.00	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	0.05	0.06
*723UnifS	73 709.00	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.02	0.07
*823UnifS	71 860.00	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.06	0.07
*923UnifS	74 100.00	✓	✓	✓	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	0.10	0.14

Cuadro 6.3: Benchmark Simulated Annealing - solución base DROP

Datos	BST	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	BST(TS)	BST(TE)
*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap113	893 076.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap121	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01

*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap123	893 076.71	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap133	893 076.71	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.01	0.02
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.02
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.02
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.02
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.02
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.00
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.00
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.02
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.01
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.00	0.01
*capa	17 156 454.48	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	0.02	0.06
*capb	12 979 071.58	✓	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓	✓	0.01	0.05
#capc	11 509 361.66	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.04	0.05
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.26
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.26
*MO3	1516.77	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.23	0.25
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.24
*MO5	1408.77	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.23	0.26
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.68	0.75
!MP2	2902.86	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.49	1.35
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		0.74
*MP4	2938.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.71	0.77
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.69	0.75
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		1.41
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		1.38
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		1.42
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		1.45
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	1.37	1.40
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.45	3.55
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.30	3.45
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	3.36	3.51
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		3.42
*MR5	2505.05	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	6.33	6.40
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓		11.70
!MT1	10 069.80	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	87.56	88.39
*1023UnifS	71 840.00	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	0.10	0.16
*1123UnifS	75 063.00	✓	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	0.11	0.13
*1223UnifS	71 756.00	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✓	0.08	0.13
*123UnifS	71 342.00	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	0.03	0.07
*1323UnifS	72 414.00	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	0.04	0.09
*1423UnifS	72 632.00	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	0.06	0.07
!1523UnifS	77 566.00	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	0.11	0.13
*1623UnifS	74 344.00	✓	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	0.06	0.06
*1723UnifS	73 847.00	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	✗	✓	0.09	0.13
*1823UnifS	71 903.00	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	0.04	0.08
*1923UnifS	75 520.00	✓	✗	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✓	✗	0.07	0.08
*2023UnifS	72 458.00	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.03	0.04
*2123UnifS	72 631.00	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.10	0.16
*2223UnifS	71 945.00	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	0.03	0.08
*223UnifS	74 148.00	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	0.03	0.06
*2323UnifS	71 335.00	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	✓	0.10	0.13
*2423UnifS	75 342.00	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✓	0.06	0.09
*2523UnifS	73 807.00	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.11	0.12
*2623UnifS	70 929.00	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	0.03	0.05
*2723UnifS	77 314.00	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	0.05	0.09
*2823UnifS	75 025.00	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	0.05	0.08
*2923UnifS	75 279.00	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	0.14	0.17
#3023UnifS	75 845.00	✗	✗	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	0.08	0.09
*323UnifS	73 319.00	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	0.12	0.14
*423UnifS	74 979.00	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.10	0.22
*523UnifS	69 251.00	✓	✓	✓	✗	✗	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	0.06	0.10
*623UnifS	72 992.00	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.05	0.07
*723UnifS	73 709.00	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.02	0.08
*823UnifS	71 860.00	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	0.04	0.05
*923UnifS	74 100.00	✓	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	0.05	0.09

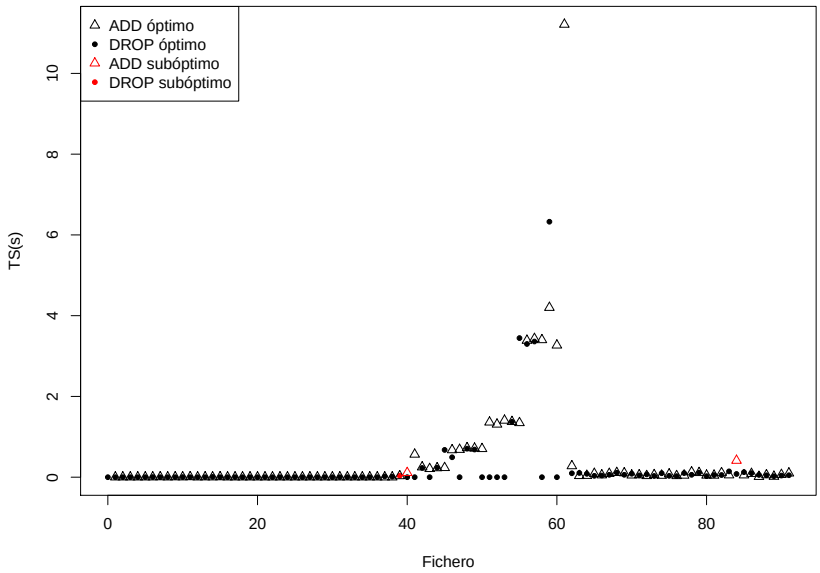

Figura 6.2: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Simulated Annealing.

Cuadro 6.4: Parámetros benchmark Threshold Accepting

Índice	T	α
0	10	0.95
1	15	0.95
2	20	0.95
3	50	0.8
4	100	0.8
5	200	0.8
6	500	0.9
7	1000	0.9

6.2. Threshold Accepting

Threshold Accepting es una metaheurística derivada de Simulated Annealing que tiene como parámetros un umbral T y una tasa de actualización α. A mayor umbral T peores soluciones se admiten. Este enfoque es similar al de Simulated Annealing, con la salvedad de que aquí no se admiten soluciones que empeoren la función objetivo según una probabilidad, sino que se toleran malas soluciones con un valor objetivo que diste del valor objetivo actual el umbral T. Cuando no se ha mejorado durante un número determinado de iteraciones (por defecto 500) se actualiza el valor umbral, de modo que se fuerce al algoritmo a realizar una búsqueda local dentro del espacio de soluciones encontrado. Aunque la filosofía va en la línea de la de Simulated Annealing, el coste computacional es menor al no calcular probabilidades, por lo que, del mismo modo, el tiempo de obtención de soluciones acostumbra a ser menor (Talbi, 2009).

Se comprueba que los tiempos son, efectivamente, mucho menores que los de Simulated Annealing, e.g. para el fichero MR5 Simulated Annealing encuentra la solución final en 6.33 segundos, frente a los 0.153 segundos de Threshold Accepting partiendo de la solución base de DROP. Una vez más, hay mayor disparidad entre los tiempos de los ficheros M* que en el resto. Llama la atención el mal comportamiento de la metaheurística con los ficheros UnifS, para los que, partiendo de cualquiera de las dos heurísticas, en el 66.66 % de estos ficheros no se llegó al óptimo. Teniendo en cuenta que son problemas más sencillos que los M*, estos resultados inducen a pensar que los parámetros evaluados no son los más adecuados para los ficheros UnifS.

6.2.1. Benchmark

Cuadro 6.5: Benchmark Threshold Accepting - solución base ADD

Datos	BST	0	1	2	3	4	5	6	7	BST(TS)	BST(TE)
*cap101	796 648.44	✗	✗	✗	✗	✗	✓	✓	✓	0.000	0.010
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.011
*cap103	893 782.11	✗	✗	✗	✗	✗	✗	✗	✓	0.001	0.012

*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.013
#cap111	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓		0.016
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap113	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓		0.019
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.022
#cap121	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓		0.016
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
*cap123	893 076.71	✗	✗	✗	✗	✗	✗	✗	✓	0.002	0.017
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.021
#cap131	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.015
*cap133	893 076.71	✗	✗	✗	✗	✗	✗	✗	✓	0.001	0.021
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.021
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap43	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
#cap51	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap63	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap73	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap81	796 648.44	✗	✗	✗	✗	✗	✓	✗	✓	0.000	0.009
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap83	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap91	796 648.44	✗	✗	✗	✗	✗	✓	✓	✗	0.000	0.009
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap93	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.488
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✗	0.037	0.318
#capc	11 526 129.40	✗	✗	✓	✗	✗	✗	✓	✗	0.086	0.261
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓	0.006	0.058
*MO2	1432.36	✗	✗	✗	✓	✓	✓	✓	✗	0.008	0.057
*MO3	1516.77	✓	✗	✓	✗	✗	✓	✓	✗	0.004	0.060
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	0.005	0.058
*MO5	1408.77	✓	✗	✓	✓	✓	✓	✓	✗	0.005	0.064
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	0.026	0.164
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	0.030	0.171
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	0.015	0.167
*MP4	2938.75	✗	✗	✗	✗	✓	✗	✓	✓	0.057	0.180
*MP5	2932.33	✓	✗	✓	✗	✓	✓	✓	✗	0.023	0.169

*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	0.061	0.330
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	0.055	0.327
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	0.041	0.326
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	0.085	0.321
*MQ5	4080.74	✓	✗	✗	✓	✓	✓	✓	✓	0.147	0.316
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	0.179	0.698
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	0.199	0.790
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	0.249	0.801
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	0.275	0.698
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	0.226	0.700
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	0.985	2.628
!MT1	10 069.80	✗	✗	✗	✗	✓	✗	✓	✓	6.604	9.739
#1023UnifS	71 963.00	✗	✗	✗	✗	✗	✗	✗	✓	0.021	0.037
#1123UnifS	75 636.00	✗	✗	✗	✗	✗	✗	✗	✓	0.027	0.037
#1223UnifS	73 096.00	✓	✓	✓	✓	✓	✓	✓	✓	0.004	0.036
*123UnifS	71 342.00	✗	✗	✗	✗	✗	✗	✗	✓	0.024	0.037
#1323UnifS	72 461.00	✓	✓	✓	✓	✗	✓	✗	✓	0.004	0.037
#1423UnifS	73 838.00	✗	✓	✓	✓	✓	✓	✓	✓	0.002	0.035
!1523UnifS	77 632.00	✗	✗	✗	✗	✗	✓	✗	✗	0.009	0.035
*1623UnifS	74 344.00	✗	✗	✗	✗	✗	✗	✗	✓	0.033	0.035
#1723UnifS	74 380.00	✓	✗	✗	✗	✗	✗	✗	✗	0.020	0.038
#1823UnifS	71 957.00	✓	✓	✓	✓	✓	✓	✓	✓	0.004	0.036
#1923UnifS	75 838.00	✗	✗	✗	✗	✗	✗	✓	✗	0.005	0.036
*2023UnifS	72 458.00	✗	✗	✗	✗	✗	✓	✗	✓	0.011	0.036
#2123UnifS	74 018.00	✗	✗	✗	✗	✗	✓	✗	✗	0.007	0.039
#2223UnifS	72 177.00	✓	✗	✓	✓	✗	✓	✓	✓	0.005	0.036
#223UnifS	74 665.00	✗	✗	✗	✗	✗	✓	✗	✓	0.002	0.037
*2323UnifS	71 335.00	✗	✗	✗	✗	✗	✗	✗	✓	0.018	0.036
#2423UnifS	76 716.00	✗	✓	✗	✗	✗	✗	✗	✗	0.006	0.035
#2523UnifS	74 149.00	✗	✗	✗	✗	✓	✗	✓	✗	0.006	0.037
*2623UnifS	70 929.00	✓	✗	✗	✗	✓	✗	✓	✗	0.002	0.037
*2723UnifS	77 314.00	✓	✓	✓	✓	✓	✓	✓	✗	0.004	0.036
#2823UnifS	75 056.00	✗	✗	✗	✗	✗	✓	✓	✗	0.003	0.036
#2923UnifS	75 577.00	✗	✗	✗	✗	✗	✓	✗	✗	0.013	0.037
#3023UnifS	75 845.00	✓	✓	✗	✓	✗	✓	✓	✗	0.002	0.036
#323UnifS	73 600.00	✗	✗	✗	✗	✗	✓	✗	✗	0.009	0.038
#423UnifS	74 984.00	✗	✗	✗	✗	✗	✗	✓	✓	0.018	0.036
*523UnifS	69 251.00	✓	✓	✓	✗	✗	✗	✗	✗	0.011	0.036
#623UnifS	73 139.00	✗	✗	✗	✓	✗	✗	✗	✓	0.011	0.036
*723UnifS	73 709.00	✗	✗	✗	✗	✗	✗	✓	✗	0.014	0.038
*823UnifS	71 860.00	✗	✓	✓	✓	✗	✗	✓	✗	0.020	0.037
#923UnifS	75 047.00	✗	✗	✗	✗	✓	✗	✗	✗	0.005	0.035

Cuadro 6.6: Benchmark Threshold Accepting - solución base ADD

Datos	BST	0	1	2	3	4	5	6	7	BST(TS)	BST(TE)
*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.011
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.012
*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.016
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap113	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.017
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.021
*cap121	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.017
*cap123	893 076.71	✗	✗	✗	✗	✗	✗	✓	✓	0.001	0.021
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.022
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
*cap133	893 076.71	✗	✗	✗	✗	✗	✗	✗	✓	0.006	0.019
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.020
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.008
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.008
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.008
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.013
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	0.051	0.487
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✓	0.094	0.317
#capc	11 535 255.51	✓	✓	✓	✓	✓	✓	✓	✓	0.029	0.268
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓		0.059
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓		0.056
*MO3	1516.77	✗	✓	✓	✗	✓	✓	✗	✗	0.004	0.060
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓		0.058

*MO5	1408.77	✓	✓	✓	✓	✓	✓	✓	✗	0.003	0.064
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	0.033	0.164
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓		0.170
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓		0.165
*MP4	2938.75	✗	✗	✗	✓	✓	✓	✗	✗	0.035	0.170
*MP5	2932.33	✓	✓	✓	✗	✓	✓	✓	✓	0.013	0.166
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓		0.323
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓		0.327
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓		0.319
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓		0.323
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✓	0.115	0.323
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	0.218	0.683
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	0.131	0.786
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	0.252	0.793
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓		0.677
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	0.153	0.686
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓		2.602
!MT1	10 069.80	✗	✗	✗	✗	✗	✓	✗	✓	7.610	9.950
#1023UnifS	72 097.00	✗	✗	✗	✓	✗	✓	✗	✗	0.002	0.036
*1123UnifS	75 063.00	✓	✓	✗	✗	✗	✗	✓	✗	0.008	0.035
#1223UnifS	72 057.00	✓	✓	✗	✓	✗	✗	✗	✗	0.007	0.036
*123UnifS	71 342.00	✗	✗	✓	✓	✗	✓	✗	✗	0.010	0.036
#1323UnifS	72 490.00	✗	✗	✓	✓	✗	✗	✗	✗	0.019	0.037
*1423UnifS	72 632.00	✗	✗	✗	✗	✗	✗	✗	✓	0.026	0.037
!1523UnifS	77 566.00	✗	✗	✗	✗	✗	✗	✓	✗	0.018	0.037
#1623UnifS	74 610.00	✗	✗	✗	✗	✗	✓	✗	✓	0.003	0.036
#1723UnifS	76 184.00	✗	✗	✗	✗	✗	✗	✗	✓	0.029	0.037
#1823UnifS	71 957.00	✓	✓	✗	✗	✗	✗	✓	✗	0.008	0.036
#1923UnifS	75 838.00	✗	✗	✗	✗	✗	✗	✓	✗	0.011	0.035
*2023UnifS	72 458.00	✗	✗	✗	✗	✓	✗	✓	✓	0.008	0.035
#2123UnifS	74 483.00	✗	✗	✗	✗	✗	✗	✗	✓	0.028	0.036
#2223UnifS	72 522.00	✗	✗	✗	✗	✗	✓	✗	✗	0.003	0.037
*223UnifS	74 148.00	✗	✗	✗	✗	✗	✗	✓	✗	0.009	0.036
#2323UnifS	71 686.00	✗	✗	✗	✗	✗	✗	✗	✓	0.004	0.036
#2423UnifS	76 754.00	✗	✗	✓	✓	✓	✓	✓	✗	0.013	0.036
*2523UnifS	73 807.00	✗	✗	✗	✗	✗	✗	✓	✗	0.014	0.035
*2623UnifS	70 929.00	✗	✗	✗	✗	✗	✗	✓	✗	0.003	0.037
#2723UnifS	78 024.00	✓	✓	✗	✗	✗	✓	✓	✗	0.001	0.037
#2823UnifS	75 114.00	✗	✗	✗	✗	✓	✗	✗	✗	0.010	0.035
#2923UnifS	75 637.00	✗	✗	✗	✗	✗	✗	✗	✓	0.012	0.037
#3023UnifS	76 557.00	✓	✓	✓	✓	✓	✗	✓	✗	0.003	0.035
#323UnifS	73 574.00	✓	✓	✓	✓	✓	✓	✓	✗	0.000	0.036
*423UnifS	74 979.00	✗	✗	✗	✗	✗	✗	✗	✓	0.012	0.036
#523UnifS	70 940.00	✓	✓	✓	✓	✓	✓	✗	✗	0.005	0.036
#623UnifS	73 139.00	✗	✗	✗	✗	✓	✓	✗	✗	0.004	0.036
#723UnifS	74 106.00	✗	✗	✗	✗	✗	✗	✓	✗	0.022	0.035
*823UnifS	71 860.00	✗	✗	✗	✗	✗	✓	✓	✗	0.002	0.037

#923UnifS	74 237.00	✗	✗	✓	✓	✓	✓	✗	✗	0.004	0.036

6.3. Late Acceptance

Late Acceptance es una metaheurística basada en Hill Climbing, una de las primeras técnicas de búsqueda local, que acepta candidatos con el mismo o mejor coste que la solución actual. Los resultados de este método suelen ser de baja calidad pero en un tiempo de cálculo bajo, por lo que era una técnica valorada cuando los procesadores no eran lo suficientemente rápidos como para que algoritmos más complicados proporcionaran buenos resultados en un tiempo practicable. La condición de aceptación de otros métodos revisados en este trabajo, como Simulated Annealing, se regula mediante un *parámetro de control* (e.g. temperatura en Simulated Annealing o umbral en Threshold Accepting) que varía a lo largo de la búsqueda. Esta forma de controlar el algoritmo permite decidir cuándo terminará, pero encontrar el valor del parámetro adecuado puede ser difícil, ya que es dependiente del problema e, incluso, puede serlo de la instancia concreta. Late Acceptance no adolece de este problema, ya que el control del algoritmo depende de los valores de la función objetivo. En concreto, esta metaheurística solo acepta una solución cuando esta es mejor que la solución de hace L_h iteraciones. L_h es el único parámetro del algoritmo (Burke y Bykov, 2012).

6.3.1. Benchmark

Late Acceptance presenta unos resultados muy similares a los de Threshold Accepting, según se percibe al comparar las figuras 6.3 y 6.4. Los resultados para los ficheros UnifS son peores aún, pues partiendo de la solución de ADD no se llega al óptimo en el 80.0 % de los casos, mientras que partiendo de la de ADD, no se alcanza en el 73.3 %.

Cuadro 6.7: Benchmark Late Acceptance - solución base ADD

Datos	**BST**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**BST(TS)**	**BST(TE)**
#cap101	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap103	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.012
#cap111	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap113	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.021
#cap121	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016

#cap123	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.022
#cap131	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.001	0.016
#cap133	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.021
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap43	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
#cap51	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap63	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap73	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.010
#cap81	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap83	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.012
#cap91	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap93	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.489
*capb	12 979 071.58	✓	✓	✗	✓	✓	✓	✓	✓	0.026	0.318
*capc	11 505 594.33	✗	✓	✗	✗	✗	✗	✗	✗	0.102	0.263
*MO1	1305.95	✗	✓	✓	✓	✗	✓	✓	✓	0.003	0.057
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓	0.003	0.056
*MO3	1516.77	✗	✓	✓	✗	✗	✗	✗	✓	0.008	0.058
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	0.002	0.058
*MO5	1408.77	✗	✓	✓	✓	✓	✗	✗	✓	0.005	0.063
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	0.029	0.164
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	0.025	0.168
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	0.030	0.167
*MP4	2938.75	✓	✗	✓	✓	✗	✗	✗	✗	0.029	0.166
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	0.014	0.167
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	0.075	0.330
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	0.039	0.327
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	0.030	0.321
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	0.051	0.321
*MQ5	4080.74	✓	✓	✓	✓	✗	✓	✗	✗	0.077	0.301
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	0.241	0.694
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	0.200	0.782

*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	0.201	0.787
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	0.218	0.691
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	0.246	0.691
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	0.393	2.599
*MT1	10 089.46	✓	✗	✗	✗	✓	✗	✗	✗	2.299	9.663
#1023UnifS	72 097.00	✗	✗	✗	✗	✗	✗	✗	✓	0.024	0.036
*1123UnifS	75 063.00	✗	✗	✗	✗	✗	✗	✗	✓	0.005	0.036
#1223UnifS	73 096.00	✓	✓	✓	✓	✓	✗	✓	✓	0.006	0.036
#123UnifS	72 359.00	✓	✓	✓	✗	✓	✓	✓	✓	0.009	0.037
#1323UnifS	72 461.00	✓	✓	✓	✓	✓	✓	✓	✓	0.008	0.037
#1423UnifS	73 625.00	✗	✗	✗	✗	✗	✓	✗	✓	0.002	0.036
#1523UnifS	78 955.00	✓	✓	✗	✓	✓	✓	✓	✓	0.001	0.035
#1623UnifS	75 529.00	✗	✗	✗	✗	✗	✗	✗	✓	0.012	0.036
#1723UnifS	74 407.00	✗	✓	✓	✗	✗	✗	✗	✗	0.009	0.037
#1823UnifS	71 957.00	✓	✗	✓	✓	✓	✗	✓	✓	0.003	0.036
#1923UnifS	76 182.00	✗	✗	✓	✓	✓	✗	✓	✓	0.006	0.035
*2023UnifS	72 458.00	✓	✓	✗	✗	✗	✓	✗	✗	0.012	0.036
#2123UnifS	74 661.00	✗	✗	✗	✗	✗	✓	✓	✓	0.007	0.036
*2223UnifS	71 945.00	✗	✗	✗	✓	✗	✗	✗	✓	0.008	0.037
#223UnifS	75 977.00	✓	✓	✓	✓	✓	✓	✓	✓		0.036
#2323UnifS	72 553.00	✓	✓	✓	✓	✓	✓	✓	✓	0.003	0.037
#2423UnifS	76 716.00	✓	✓	✓	✗	✓	✓	✓	✗	0.012	0.034
#2523UnifS	74 149.00	✗	✗	✗	✗	✗	✗	✗	✓	0.013	0.036
*2623UnifS	70 929.00	✓	✓	✓	✗	✗	✗	✓	✓	0.005	0.036
*2723UnifS	77 314.00	✓	✓	✓	✗	✓	✓	✗	✓	0.009	0.036
#2823UnifS	75 056.00	✗	✗	✗	✗	✗	✓	✓	✗	0.004	0.036
#2923UnifS	75 763.00	✗	✓	✓	✓	✗	✓	✗	✗	0.007	0.035
#3023UnifS	75 845.00	✓	✓	✓	✓	✗	✗	✗	✓	0.006	0.035
#323UnifS	73 547.00	✗	✓	✓	✓	✗	✗	✗	✗	0.007	0.038
#423UnifS	79 414.00	✓	✓	✓	✓	✓	✓	✓	✓		0.035
#523UnifS	70 410.00	✗	✗	✗	✗	✗	✗	✗	✓	0.002	0.036
#623UnifS	73 139.00	✓	✓	✓	✗	✓	✓	✓	✗	0.002	0.035
#723UnifS	74 406.00	✓	✗	✗	✗	✓	✓	✓	✓	0.002	0.035
*823UnifS	71 860.00	✗	✓	✓	✓	✓	✓	✓	✓	0.008	0.037
#923UnifS	74 571.00	✗	✗	✗	✓	✓	✓	✗	✗	0.010	0.035

Cuadro 6.8: Benchmark Late Acceptance - solución base DROP

Datos	**BST**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**BST(TS)**	**BST(TE)**
*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.011
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.012

*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.016
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap113	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.017
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.021
*cap121	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap123	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.021
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.017
#cap133	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.017
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.021
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.008
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.008
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.008
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.012
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	0.025	0.487
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✓	0.095	0.317
#capc	11 535 255.51	✓	✓	✓	✓	✓	✓	✓	✓	0.041	0.268
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓		0.057
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓		0.055
#MO3	1521.47	✓	✓	✓	✓	✓	✓	✓	✓		0.056
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓		0.057
#MO5	1411.11	✓	✓	✓	✓	✓	✓	✓	✓		0.063
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	0.010	0.164
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓		0.166
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓		0.165
#MP4	2945.65	✓	✓	✓	✓	✓	✓	✓	✓		0.164
#MP5	2939.95	✓	✓	✓	✓	✓	✓	✓	✓		0.165
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓		0.321

*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓		0.326
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓		0.318
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓		0.322
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✗	0.094	0.322
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	0.056	0.679
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	0.230	0.786
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	0.108	0.791
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓		0.678
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	0.144	0.685
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓		2.599
*MT1	10 089.46	✗	✗	✗	✗	✓	✓	✗	✓	4.256	9.613
*1023UnifS	71 840.00	✗	✗	✗	✓	✓	✗	✗	✗	0.014	0.036
#1123UnifS	75 352.00	✗	✗	✗	✓	✗	✗	✗	✗	0.036	0.037
#1223UnifS	72 057.00	✗	✓	✓	✓	✗	✗	✓	✓	0.010	0.035
*123UnifS	71 342.00	✓	✓	✓	✗	✗	✗	✗	✗	0.018	0.036
#1323UnifS	72 490.00	✗	✗	✗	✗	✗	✗	✓	✗	0.011	0.039
#1423UnifS	73 606.00	✗	✗	✗	✗	✗	✗	✗	✓	0.014	0.035
!1523UnifS	77 688.00	✓	✓	✗	✗	✗	✗	✓	✓	0.001	0.035
#1623UnifS	74 671.00	✓	✓	✗	✗	✗	✗	✗	✗	0.015	0.035
#1723UnifS	78 364.00	✓	✓	✓	✓	✓	✓	✓	✓		0.034
#1823UnifS	71 957.00	✓	✓	✓	✗	✗	✗	✗	✗	0.014	0.036
#1923UnifS	75 838.00	✓	✓	✓	✓	✓	✓	✓	✗	0.001	0.035
#2023UnifS	73 765.00	✓	✓	✗	✗	✗	✓	✓	✗	0.008	0.035
*2123UnifS	72 631.00	✗	✗	✗	✗	✓	✗	✗	✗	0.012	0.038
#2223UnifS	72 177.00	✗	✗	✗	✗	✓	✓	✗	✓	0.020	0.036
#223UnifS	75 268.00	✗	✗	✓	✓	✓	✗	✓	✓	0.004	0.036
#2323UnifS	75 693.00	✓	✓	✓	✓	✓	✓	✓	✓		0.035
#2423UnifS	76 754.00	✓	✓	✗	✗	✗	✓	✓	✓	0.009	0.035
#2523UnifS	74 041.00	✗	✗	✗	✗	✗	✓	✗	✓	0.010	0.036
#2623UnifS	71 228.00	✓	✓	✓	✓	✓	✓	✓	✓	0.003	0.036
*2723UnifS	77 314.00	✗	✗	✗	✗	✗	✗	✗	✓	0.005	0.035
#2823UnifS	76 839.00	✓	✓	✓	✓	✓	✓	✓	✓		0.034
#2923UnifS	75 409.00	✗	✗	✗	✗	✓	✓	✓	✗	0.008	0.036
#3023UnifS	76 557.00	✓	✓	✗	✓	✓	✓	✓	✓	0.005	0.034
#323UnifS	73 574.00	✓	✓	✓	✗	✓	✓	✗	✗	0.004	0.035
#423UnifS	75 186.00	✓	✓	✓	✓	✓	✗	✓	✓	0.007	0.035
*523UnifS	69 251.00	✗	✗	✗	✓	✓	✓	✓	✓	0.011	0.036
*623UnifS	72 992.00	✗	✗	✗	✗	✗	✓	✓	✗	0.012	0.036
#723UnifS	74 733.00	✓	✓	✓	✓	✓	✓	✓	✓		0.036
*823UnifS	71 860.00	✗	✗	✗	✗	✓	✓	✗	✗	0.014	0.037
#923UnifS	74 237.00	✓	✓	✓	✗	✓	✓	✗	✗	0.002	0.036

Algorithm 8 Threshold Accepting

function THRESHOLDACCEPTANCE
    Sea $z_{mejor} = \infty$ el mejor valor objetivo encontrado hasta el momento
    Sea z_n el valor objetivo de la iteración actual
    Sea z_{cand} el valor objetivo de la solución candidata de la búsqueda local
    Sea S_n la solución de la iteración actual
    Sea S_{cand} la solución candidata de la búsqueda local
    Sea S_{mejor} la mejor solución encontrada
    Se inicializa S_{mejor} y z_{mejor}
    Se copia S_{mejor} a S_n
    while $fin = 0$ **do** ▷ Iterar hasta que se cumpla la condición de terminación
        $busquedaLocal(S_n)$ ▷ S_{cand} guarda el movimiento
        $z_{cand} = valorObjetivo(S_{cand})$
        $\delta = z_{cand} - z_n$
        if $\delta < T$ **then**
            $S_n = S_{cand}$
            $z_n = z_{cand}$
            if $z_n < z_{mejor}$ **then**
                $S_{mejor} = S_n$
                $z_{mejor} = z_n$
            end if
        end if
        $actualizarUmbral()$
    end while
end function

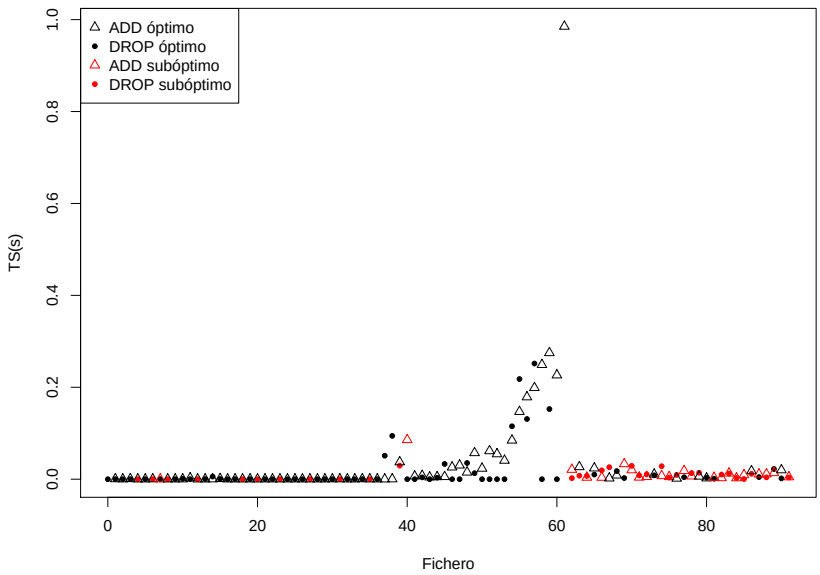

Figura 6.3: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Threshold Accepting.

Algorithm 9 Late Acceptance

```
1: function LATEACCEPTANCE
2:     Sea z_mejor = ∞ el mejor valor objetivo encontrado hasta el momento
3:     Sea z_n el valor objetivo de la iteración actual
4:     Sea z_cand el valor objetivo de la solución candidata de la búsqueda local
5:     Sea S_n la solución de la iteración actual
6:     Sea S_cand la solución candidata de la búsqueda local
7:     Sea S_mejor la mejor solución encontrada
8:     Se inicializa S_mejor y z_mejor
9:     Se copia S_mejor a S_n
10:    Sea fitness[1..Lh] el array de valores objetivo de las últimas L_h iteraciones
11:    Sea v el índice virtual de fitness
12:    while fin = 0 do          ▷ Iterar hasta que se cumpla la condición de terminación
13:        busquedaLocal(S_n)                          ▷ S_cand guarda el movimiento
14:        z_cand = valorObjetivo(S_cand)
15:        v = it %L_h                                 ▷ it corresponde a la iteración actual
16:        if z_cand < fitness[v] ∨ z_cand ≤ z_n then
17:            z_n = z_cand
18:            S_n = S_cand
19:            z_n = z_cand
20:            if z_n < z_mejor then
21:                S_mejor = S_n
22:                z_mejor = z_n
23:            end if
24:        end if
25:        if z_n < fitness[v] then
26:            fitness[v] = z_n
27:        end if
28:    end while
29: end function
```

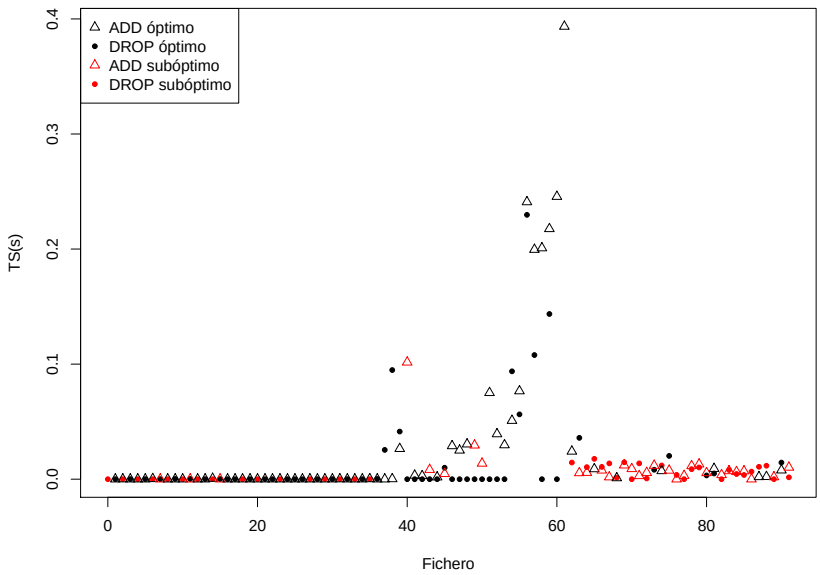

Figura 6.4: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Late Acceptance.

6.4. Old Bachelor

Old Bachelor tiene, al igual que Threshold Accepting, un umbral de aceptación T. Sin embargo, en lugar de disminuir el umbral cuando no se ha mejorado el valor objetivo durante un número determinado de iteraciones Old Bachelor aumenta el umbral a medida que encuentra malas soluciones (de ahí el nombre del algoritmo, pues el viejo solterón baja sus espectativas [aumenta el umbral de candidatas] a medida que es rechazado). De este modo el algoritmo puede escapar de mínimos locales en los que Threshold Accepting se quedaría atascado, como en situaciones en las que el valor objetivo actual es mucho menor que el de todos los vecinos. Del mismo modo, el solterón se vuelve más exigente (T disminuye) a medida que se encuentran soluciones que mejoran el valor objetivo actual. En el cuadro algorítmico 10 se muestra pseudocódigo de la metaheurística Old Bachelor.

En este trabajo se han evaluado dos variantes de Old Bachelor, que se han denominado Old Bachelor 1 (OBA1) y Old Bachelor 2 (OBA2), como en el artículo original (Hu, Kahng y Tsao, 1995). La versión OBA1 tiene los siguientes parámetros:

- a: cambia la tasa de crecimiento del umbral con un factor multiplicativo.
- b: permite un crecimiento potencial.
- c modifica el factor de damping $(1 - \frac{i}{M})$, usado para escalar la magnitud del umbral según el algoritmo se acerca a las iteraciones finales.
- Δ: granularidad de actualización del umbral.

La actualización del umbral viene determinada por la siguiente igualdad:

$$T_{i+1} = \left(\left(\frac{age}{a}\right)^b - 1\right) * (\Delta) * \left(1 - \frac{i}{M}\right)^c \tag{6.1}$$

Nótese que el nuevo valor umbral T_{i+1} no es función del valor previo T_i, sino que viene completamente determinado por la edad (age), la iteración actual i y la parametrización del algoritmo.

Cuadro 6.9: Parámetros benchmark Old Bachelor 1

Datos	**a**	**b**	**c**	**Δ**	**M**
0	10	1	0.5	0.093	10000
1	10	2	0.5	0.093	10000
2	10	3	0.5	0.093	10000
3	10	4	0.5	0.093	10000
4	10	5	0.5	0.093	10000
5	50	1	0.5	0.093	10000
6	50	2	0.5	0.093	10000

7	50	3	0.5	0.093	10000
8	50	4	0.5	0.093	10000
9	50	5	0.5	0.093	10000
10	100	1	0.5	0.093	10000
11	100	2	0.5	0.093	10000
12	100	3	0.5	0.093	10000
13	100	4	0.5	0.093	10000
14	100	5	0.5	0.093	10000
15	500	1	0.5	0.093	10000
16	500	2	0.5	0.093	10000
17	500	3	0.5	0.093	10000
18	500	4	0.5	0.093	10000
19	500	5	0.5	0.093	10000
20	1000	1	0.5	0.093	10000
21	1000	2	0.5	0.093	10000
22	1000	3	0.5	0.093	10000
23	1000	4	0.5	0.093	10000
24	1000	5	0.5	0.093	10000

6.4.1. Benchmark

Old Bachelor 1 alcanza los mejores resultados en tiempo de las metaheurísticas analizadas en este trabajo. Los tiempos de Old Bachelor 2 son también muy buenos. Para ambas versiones de la metaheurística se llega a la solución final en menos de 0.2 segundos. Existe el mismo problema con los ficheros UnifS que con las metaheurísticas Threshold Accepting y Late Acceptance. En el caso de Old Bachelor 1 no se llega al óptimo en el 80 % de los ficheros UnifS partiendo de ADD y el 76 % desde DROP. Old Bachelor 2, por otro lado, no alcanza el óptimo en el 76 %, independientemente de la heurística utilizada.

Cuadro 6.10: Benchmark Old Bachelor 1 - solución base ADD

Datos	BST	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	BST(TS)	BST
#cap101	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.
#cap103	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
#cap111	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.
#cap113	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
#cap121	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.
#cap123	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
#cap131	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.
#cap133	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.
#cap43	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.
#cap51	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.

*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.008
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000		0.008
#cap63	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.010
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.008
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000		0.008
#cap73	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.010
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.010
#cap81	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.010
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000		0.010
#cap83	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.012
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.013
#cap91	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.010
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000		0.010
#cap93	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.012
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.013
*capa	17 156 454.48	✓	✓	✓	✓	✓44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000		0.488
*capb	12 979 071.58	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	0.017		0.317
*capc	11 505 594.33	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	0.071		0.260
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.002		0.058
*MO2	1432.36	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.002		0.055
*MO3	1516.77	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	0.001		0.060
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000		0.058
*MO5	1408.77	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	0.002		0.064
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.016		0.164
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.026		0.167
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009		0.166
*MP4	2938.75	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	0.017		0.167
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	0.007		0.165
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.043		0.328
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.029		0.326
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.021		0.321
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.017		0.322
*MQ5	4080.74	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✗	0.071		0.304
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.154		0.686
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.056		0.786
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.171		0.791
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.134		0.681
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.097		0.686
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.587		2.599
!MT1	10 069.80	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	5.515		9.686
#1023UnifS	72 097.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	0.015		0.036
#1123UnifS	75 265.00	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.004		0.036
#1223UnifS	73 096.00	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.003		0.037
#123UnifS	72 359.00	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.004		0.037
#1323UnifS	72 461.00	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	0.002		0.038
#1423UnifS	73 838.00	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.001		0.035
!1523UnifS	77 579.00	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	✓	0.012		0.035
#1623UnifS	74 540.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.014		0.037
#1723UnifS	74 407.00	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.010		0.037
#1823UnifS	71 957.00	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.003		0.037
#1923UnifS	76 182.00	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.016		0.037
*2023UnifS	72 458.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	0.006		0.037
#2123UnifS	74 661.00	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.006		0.036
#2223UnifS	72 177.00	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	0.005		0.037
#223UnifS	75 977.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.037
#2323UnifS	72 553.00	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.001		0.037
#2423UnifS	76 343.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	0.011		0.036
#2523UnifS	74 149.00	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	0.004		0.036
*2623UnifS	70 929.00	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	0.006		0.036
*2723UnifS	77 314.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.004		0.036
#2823UnifS	75 056.00	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.009		0.036
#2923UnifS	75 577.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.006		0.037
#3023UnifS	75 845.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.001		0.036
#323UnifS	73 623.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.016		0.037
#423UnifS	79 414.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			0.036
*523UnifS	69 251.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.013		0.037
#623UnifS	73 139.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	0.002		0.036
#723UnifS	74 406.00	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	0.001		0.035
*823UnifS	71 860.00	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	0.002		0.038
#923UnifS	74 571.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	0.008		0.035

Cuadro 6.11: Benchmark Old Bachelor 1 - solución base DROP

Datos	BST	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	BST(TS)	BST(TE)
*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.011
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.013
*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap113	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.021
*cap121	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap123	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.018

*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.02
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
#cap133	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.02
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.00
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.00
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.00
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.00
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.00
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.00
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.01
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.01
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.024	0.48
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.074	0.31
#capc	11 535 255.51	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010	0.26
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.05
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.05
#MO3	1521.47	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.05
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.05
#MO5	1411.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.06
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.007	0.16
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.16
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.16
#MP4	2945.65	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.16
#MP5	2939.95	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.16
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.32
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.32
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.31
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.32
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.105	0.32
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.087	0.68
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.124	0.78
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.121	0.79
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.67
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.084	0.68
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		2.59
*MT1	10 089.46	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓	✓	2.582	9.63
#1023UnifS	72 364.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	0.010	0.03
#1123UnifS	75 312.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.018	0.03
#1223UnifS	72 057.00	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✓	0.003	0.03
*123UnifS	71 342.00	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	0.008	0.03
#1323UnifS	72 942.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.007	0.03
#1423UnifS	73 516.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✓	0.008	0.03
!1523UnifS	77 688.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	0.001	0.03
#1623UnifS	74 671.00	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.022	0.03
#1723UnifS	78 364.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.03
#1823UnifS	71 957.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.026	0.03
#1923UnifS	75 838.00	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	0.001	0.03
*2023UnifS	72 458.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.009	0.03
#2123UnifS	73 379.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	0.021	0.03
*2223UnifS	71 945.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.008	0.03
#223UnifS	75 112.00	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.006	0.03
#2323UnifS	75 693.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.03
#2423UnifS	76 592.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.027	0.03
#2523UnifS	73 926.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.026	0.03
*2623UnifS	70 929.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	0.006	0.03
*2723UnifS	77 314.00	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.009	0.03
#2823UnifS	76 839.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.03
#2923UnifS	75 577.00	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✗	0.003	0.03
#3023UnifS	76 557.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	0.001	0.03
#323UnifS	73 574.00	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✗	✓	0.001	0.03
#423UnifS	75 186.00	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	0.004	0.03
*523UnifS	69 251.00	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	0.006	0.03
#623UnifS	73 139.00	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.011	0.03
#723UnifS	74 733.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.03
#823UnifS	71 949.00	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.002	0.03
#923UnifS	74 237.00	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	0.000	0.03

Cuadro 6.12: Benchmark Old Bachelor 2 - solución base ADD

Datos	BST	0	1	2	3	4	5	6	7	8	9	BST(TS)	BST(TE)
#cap101	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap103	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.012
#cap111	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap113	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.021
#cap121	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
#cap123	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.021
#cap131	794 299.85	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.001	0.016
#cap133	894 095.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.018
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.021
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap43	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
#cap51	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap63	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.008
#cap73	1 012 476.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
#cap81	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap83	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.012
#cap91	797 508.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.009
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
#cap93	895 027.19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.488
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.017	0.318
*capc	11 505 594.33	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗	0.073	0.254
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.004	0.057
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.005	0.056
*MO3	1516.77	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.006	0.057
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.057
*MO5	1408.77	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	0.002	0.063
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.030	0.164
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.034	0.169
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.021	0.166
*MP4	2938.75	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	0.053	0.168
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.012	0.167

*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.065	0.328
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.037	0.326
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.033	0.319
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.057	0.321
*MQ5	4080.74	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗	0.061	0.311
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.154	0.693
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.183	0.789
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.242	0.790
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.108	0.682
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.041	0.688
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.765	2.603
!MT1	10 069.80	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	5.051	9.678
#1023UnifS	72 097.00	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	0.022	0.035
#1123UnifS	75 571.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.011	0.035
#1223UnifS	73 096.00	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	0.006	0.035
#123UnifS	72 359.00	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	0.005	0.037
#1323UnifS	72 461.00	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	0.001	0.037
#1423UnifS	73 838.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.002	0.036
!1523UnifS	77 579.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.030	0.037
*1623UnifS	74 344.00	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.009	0.035
#1723UnifS	74 407.00	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.012	0.037
#1823UnifS	71 957.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.004	0.036
#1923UnifS	76 259.00	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	0.012	0.036
#2023UnifS	74 138.00	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	0.004	0.036
#2123UnifS	74 661.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.027	0.036
*2223UnifS	71 945.00	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	0.030	0.036
#223UnifS	75 977.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.036
#2323UnifS	72 553.00	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	0.001	0.036
#2423UnifS	76 793.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.011	0.035
#2523UnifS	74 149.00	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	0.003	0.036
*2623UnifS	70 929.00	✓	✓	✓	✗	✗	✓	✓	✓	✗	✗	0.005	0.035
*2723UnifS	77 314.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.007	0.035
#2823UnifS	75 056.00	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	0.006	0.036
#2923UnifS	75 763.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.002	0.036
#3023UnifS	75 845.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.006	0.035
*323UnifS	73 319.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.026	0.036
#423UnifS	79 414.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.035
*523UnifS	69 251.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.006	0.036
#623UnifS	73 800.00	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	0.003	0.035
#723UnifS	75 254.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.005	0.035
#823UnifS	71 977.00	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.017	0.038
#923UnifS	74 571.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.015	0.034

Cuadro 6.13: Benchmark Old Bachelor 2 - solución base DROP

Datos	**BST**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**BST(TS)**	**BST(TE)**

*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.011
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.012
*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.015
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.016
#cap113	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.017
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.021
*cap121	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.016
#cap123	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.017
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.021
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.016
#cap133	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.017
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.021
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.008
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.008
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.008
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.011
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.012
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.009
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.042
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.084
#capc	11 535 255.51	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.022
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.057
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.055
*MO3	1516.77	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	0.020
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.059
*MO5	1408.77	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.058
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	0.026
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.063
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.010
*MP4	2938.75	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.093
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.050
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.322

*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.326
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.319
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.321
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.097	0.322
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.110	0.680
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.147	0.785
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.176	0.793
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.678
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.074	0.687
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		2.597
*MT1	10 089.46	✗	✓	✓	✓	✗	✗	✗	✗	✓	✓	4.000	9.651
*1023UnifS	71 840.00	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.008	0.035
#1123UnifS	75 331.00	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.002	0.035
#1223UnifS	72 057.00	✗	✗	✓	✗	✗	✗	✓	✓	✓	✗	0.008	0.035
*123UnifS	71 342.00	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	0.006	0.035
#1323UnifS	73 281.00	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.036	0.037
#1423UnifS	73 561.00	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	0.023	0.036
!1523UnifS	77 566.00	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	0.023	0.035
#1623UnifS	76 494.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.002	0.035
#1723UnifS	77 979.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.031	0.037
#1823UnifS	72 837.00	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.015	0.036
#1923UnifS	75 838.00	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	0.005	0.035
*2023UnifS	72 458.00	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	0.006	0.035
#2123UnifS	74 576.00	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	0.007	0.034
#2223UnifS	74 071.00	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	0.010	0.035
*223UnifS	74 148.00	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	0.027	0.036
#2323UnifS	75 693.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.035
#2423UnifS	76 754.00	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	0.017	0.035
#2523UnifS	74 041.00	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.009	0.037
*2623UnifS	70 929.00	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.003	0.037
#2723UnifS	77 997.00	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.023	0.036
#2823UnifS	75 114.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.025	0.038
#2923UnifS	75 577.00	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	0.015	0.036
#3023UnifS	76 557.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.003	0.034
#323UnifS	73 574.00	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.003	0.036
#423UnifS	75 186.00	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	0.011	0.035
*523UnifS	69 251.00	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.014	0.036
#623UnifS	73 139.00	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	0.006	0.036
#723UnifS	74 733.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.035
#823UnifS	71 949.00	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	0.003	0.036
#923UnifS	74 237.00	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	0.002	0.036

6.5. Demon Algorithm

Demon Algorithm o *Algoritmo del Demonio* es una metaheurística que se basa en el concepto de energía de un sistema. En su forma original su objetivo no es el de generar

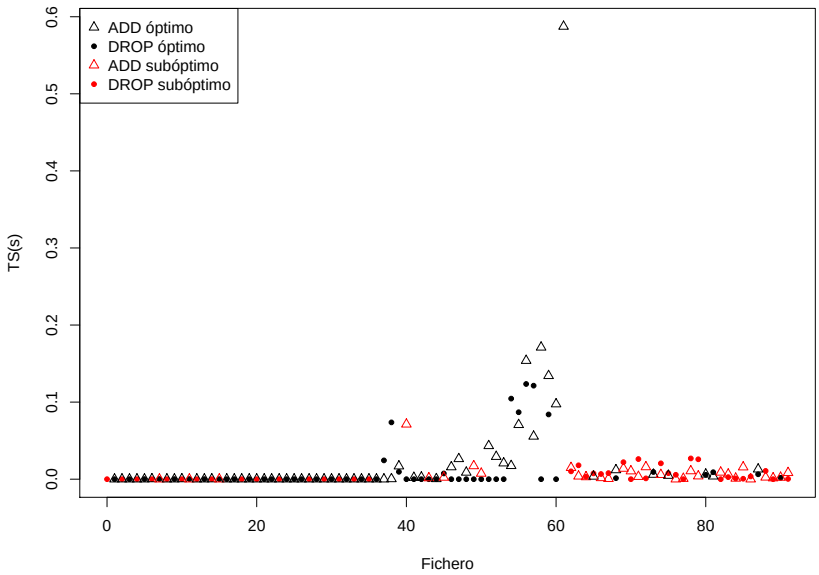

Figura 6.5: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Old Bachelor 1.

Algorithm 10 Old Bachelor

```
1: function oldBachelor
2:     Sea z_mejor = ∞ el mejor valor objetivo encontrado hasta el momento
3:     Sea z_n el valor objetivo de la iteración actual
4:     Sea z_cand el valor objetivo de la solución candidata de la búsqueda local
5:     Sea S_n la solución de la iteración actual
6:     Sea S_cand la solución candidata de la búsqueda local
7:     Sea S_mejor la mejor solución encontrada
8:     Se inicializa S_mejor y z_mejor
9:     Se copia S_mejor a S_n
10:    while fin = 0 do          ▷ Iterar hasta que se cumpla la condición de terminación
11:        busquedaLocal(S_n)                          ▷ S_cand guarda el movimiento
12:        z_cand = valorObjetivo(S_cand)
13:        if z_cand < z_n + T then
14:            z_n = z_cand
15:            S_n = S_cand
16:            z_n = z_cand
17:            if z_n < z_mejor then
18:                S_mejor = S_n
19:                z_mejor = z_n
20:            end if
21:        end if
22:    end while
23:    actualizarT()
24: end function
```

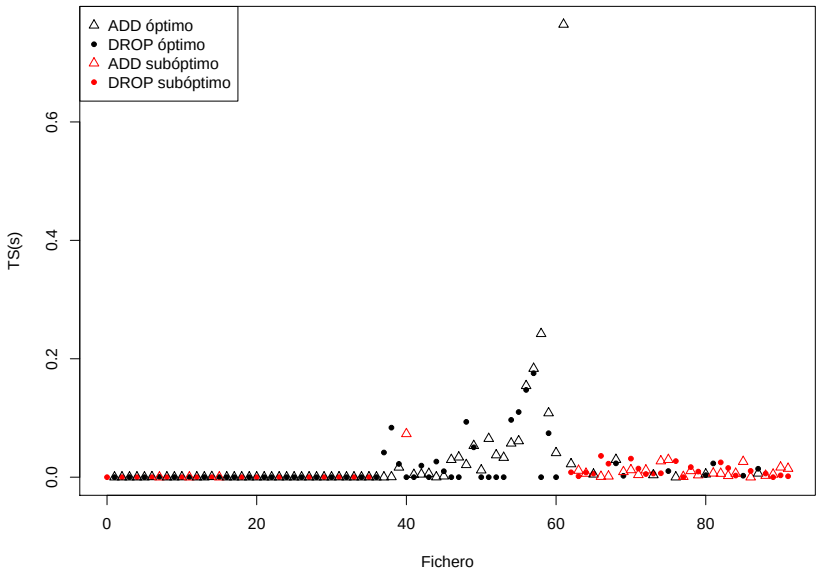

Figura 6.6: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Old Bachelor 2.

Cuadro 6.14: Parámetros benchmark Demon Algorithm

Índice	D
0	5
1	10
2	30
3	50
4	100
5	200
6	500
7	1000
8	2000

estados de baja energía, por lo que no es útil de forma directa para optimización (Wood y Downs, 1998). En este trabajo, sin embargo, se ha decidido evaluar esta forma original, ideada por Michael Creutz, para comprobar esta afirmación.

En este algoritmo, la energía perdida por el sistema se cede a una variable artificial denominada "demonio". Los incrementos en la energía del sistema solo son permitidos si el demonio puede suministrar la energía necesaria, que, como consecuencia, pierde. Como resultado, la energía total del sistema es una constante: $E(S) + D = C$ para cualquier estado en la cadena de Markov. La temperatura no se especifica directamente, pero puede ser estimada a partir de la cadena de estados. Su valor está gobernado por la energía total C, que equivale a la energía inicial más la energía inicial del demonio.

La función de aceptación para el método de Creutz es computacionalmente más simple que la de Simulated Annealing, por lo que se esperan tiempos de solución más bajos que con esta metaheurística.

En el cuadro algorítmico 11 se muestra el pseudocódigo de la metaheurística.

6.5.1. Benchmark

Demon algorithm presenta tiempos muy buenos, de menos de 1 segundo. Mejora con respecto a Threshold Accepting, Late Acceptance y Old Bachelor (en ambas versiones) en que alcanza el óptimo para todos los ficheros cap, excepto para capb (solo lo alcanza partiendo de ADD) y capc. Sin embargo, la metaheurísticas mencionadas sí alcanzan el óptimo para cab, y Old Bachelor también lo alcanza para capc (en ambas versiones partiendo de ADD). Como contrapartida, esta metaheurística es la que peores resultados da para los ficheros UnifS: no alcanza el óptimo partiendo de ADD en el 83.33 % de los ficheros y en el 86.67 % desde DROP.

Cuadro 6.15: Benchmark Demon Algorithm ADD

Datos	BST	0	1	2	3	4	5	6	7	8	BST(TS)	BST(TE)

*cap101	796 648.44	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.000	0.010
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap103	893 782.11	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.001	0.012
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap111	793 439.56	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.000	0.015
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
*cap113	893 076.71	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.002	0.017
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.021
*cap121	793 439.56	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.000	0.015
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.001	0.016
*cap123	893 076.71	✗	✗	✗	✗	✗	✗	✓	✓	✗	0.002	0.018
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.021
*cap131	793 439.56	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.001	0.015
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.016
*cap133	893 076.71	✗	✗	✗	✗	✗	✗	✓	✗	✗	0.003	0.017
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.021
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.009
*cap43	1 010 641.45	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.000	0.011
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap51	1 010 641.45	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.000	0.011
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.009
*cap63	1 010 641.45	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.000	0.011
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.007
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.009
*cap73	1 010 641.45	✗	✗	✗	✗	✗	✗	✗	✗	✓	0.000	0.011
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.010
*cap81	796 648.44	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.000	0.010
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap83	893 782.11	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.001	0.012
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap91	796 648.44	✗	✗	✗	✗	✗	✓	✓	✓	✓	0.000	0.010
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.010
*cap93	893 782.11	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.001	0.011
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.480
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.106	0.317
#capc	11 509 361.66	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.108	0.294
#MO1	1331.19	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.000	0.072
#MO2	1454.07	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.050	0.069
#MO3	1544.63	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.063	0.073
#MO4	1470.45	✗	✓	✓	✗	✗	✗	✓	✗	✗	0.007	0.071
#MO5	1423.93	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.025	0.075
#MP1	2764.85	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.123	0.215

#MP2	2961.55	✗	✓	✗	✗	✓	✗	✗	✗	✗	0.089	0.222
#MP3	2695.40	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.026	0.218
#MP4	2990.59	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.046	0.225
#MP5	2962.39	✗	✗	✗	✗	✗	✗	✗	✓	✗	0.158	0.361
#MQ1	4273.48	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.257	0.458
#MQ2	4169.64	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.097	0.439
*MQ3	4275.43	✓	✓	✓	✓	✓	✗	✗	✗	✗	0.093	0.377
*MQ4	4235.15	✓	✓	✓	✓	✓	✗	✗	✗	✗	0.054	0.322
*MQ5	4080.74	✗	✓	✓	✓	✓	✗	✗	✗	✗	0.100	0.337
#MR1	2672.60	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.182	0.909
#MR2	2763.85	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.229	0.959
#MR3	2798.04	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.139	0.957
#MR4	2801.86	✓	✓	✓	✓	✓	✗	✗	✗	✗	0.057	0.913
#MR5	2521.73	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.276	0.906
*MS1	5283.76	✓	✓	✓	✓	✓	✗	✗	✗	✗	0.446	2.609
!MT1	10 069.80	✓	✗	✓	✗	✗	✗	✗	✗	✗	5.083	9.682
#1023UnifS	72 484.00	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.008	0.038
#1123UnifS	77 037.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.031	0.038
#1223UnifS	72 092.00	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.005	0.038
#123UnifS	71 674.00	✗	✗	✗	✗	✓	✗	✗	✓	✗	0.015	0.038
*1323UnifS	72 414.00	✗	✗	✗	✗	✗	✗	✓	✗	✗	0.012	0.039
#1423UnifS	73 443.00	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.001	0.037
!1523UnifS	77 566.00	✗	✗	✗	✓	✗	✓	✗	✗	✗	0.015	0.036
#1623UnifS	74 540.00	✗	✗	✗	✗	✗	✗	✗	✓	✗	0.004	0.038
#1723UnifS	74 483.00	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.022	0.039
#1823UnifS	72 509.00	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.035	0.038
#1923UnifS	75 919.00	✗	✗	✗	✗	✗	✗	✓	✗	✗	0.030	0.037
#2023UnifS	75 352.00	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.011	0.038
*2123UnifS	72 631.00	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.031	0.038
#2223UnifS	73 311.00	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.038	0.039
#223UnifS	74 287.00	✗	✗	✗	✗	✗	✗	✓	✓	✗	0.019	0.038
#2323UnifS	71 820.00	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.008	0.038
#2423UnifS	77 435.00	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.009	0.037
#2523UnifS	74 651.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.010	0.038
#2623UnifS	71 228.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.012	0.038
#2723UnifS	77 500.00	✗	✓	✓	✗	✗	✗	✗	✗	✗	0.007	0.038
#2823UnifS	75 508.00	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.006	0.037
#2923UnifS	75 576.00	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.011	0.038
#3023UnifS	75 845.00	✓	✓	✓	✗	✗	✓	✗	✗	✗	0.016	0.037
#323UnifS	75 013.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.030	0.039
#423UnifS	76 355.00	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.019	0.036
#523UnifS	69 632.00	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.020	0.039
#623UnifS	73 468.00	✗	✗	✗	✓	✗	✓	✗	✗	✗	0.010	0.038
*723UnifS	73 709.00	✗	✗	✓	✓	✗	✓	✗	✗	✗	0.005	0.038
*823UnifS	71 860.00	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.012	0.039
#923UnifS	74 858.00	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.019	0.038

Algorithm 11 Demon Algorithm

```
1: function DEMONALGORITHM
2:     Sea z_mejor = ∞ el mejor valor objetivo encontrado hasta el momento
3:     Sea z_n el valor objetivo de la iteración actual
4:     Sea z_cand el valor objetivo de la solución candidata de la búsqueda local
5:     Sea S_n la solución de la iteración actual
6:     Sea S_cand la solución candidata de la búsqueda local
7:     Sea S_mejor la mejor solución encontrada
8:     Se inicializa S_mejor y z_mejor
9:     Se copia S_mejor a S_n
10:    Sea D la energía del demonio
11:    while fin = 0 do                ▷ Iterar hasta que se cumpla la condición de terminación
12:        busquedaLocal(S_n)                          ▷ S_cand guarda el movimiento
13:        z_cand = valorObjetivo(S_cand)
14:        δ = z_cand − z_n
15:        if δ ≤ D then
16:            z_n = z_cand
17:            S_n = S_cand
18:            z_n = z_cand
19:            D = D − δ
20:            if z_n < z_mejor then
21:                S_mejor = S_n
22:                z_mejor = z_n
23:            end if
24:        end if
25:    end while
26: end function
```

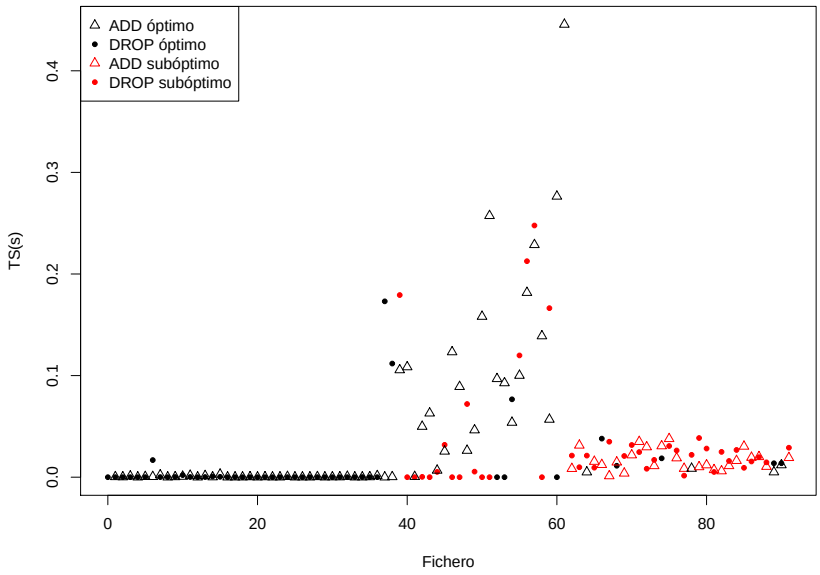

Figura 6.7: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Demon Algorithm.

Cuadro 6.16: Benchmark Demon Algorithm DROP

Datos	BST	0	1	2	3	4	5	6	7	8	BST(TS)	BST(TE)
*cap101	796 648.44	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.004
*cap102	854 704.20	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.000	0.003
*cap103	893 782.11	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.003
!cap111	793 439.56	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.017
*cap111	793 439.56	✗	✓	✓	✗	✗	✗	✗	✗	✗		0.014
*cap112	851 495.33	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.000	0.013
*cap113	893 076.71	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.017	0.032
!cap121	793 439.56	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.014
#cap122	851 495.33	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.000	0.013
#cap123	893 076.71	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.000	0.015
!cap131	793 439.56	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.018
!cap131	793 439.56	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.012
#cap132	851 495.33	✗	✓	✓	✗	✗	✗	✗	✗	✗	0.000	0.013
#cap133	893 076.71	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.000	0.012
#cap134	928 941.75	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.006	0.020
#cap41	932 615.75	✗	✗	✓	✗	✗	✗	✗	✗	✗		0.001
#cap43	1 010 641.45	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.001
#cap51	1 010 641.45	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.004
!cap61	932 615.75	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.004
!cap61	932 615.75	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.001
!cap62	977 799.40	✗	✓	✓	✗	✗	✗	✗	✗	✗		0.001
#cap63	1 010 641.45	✗	✗	✓	✗	✗	✗	✗	✗	✗		0.001
!cap71	932 615.75	✓	✓	✗	✗	✗	✗	✗	✗	✗		0.002
!cap71	932 615.75	✗	✗	✓	✗	✗	✗	✗	✗	✗		0.001
!cap73	1 010 641.45	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.001
!cap81	796 648.44	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.007
!cap81	796 648.44	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.003
!cap82	854 704.20	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.000	0.004
!cap83	893 782.11	✗	✓	✓	✗	✗	✗	✗	✗	✗		0.003
*cap91	796 648.44	✓	✓	✗	✗	✗	✗	✗	✗	✗		0.006
!cap91	796 648.44	✓	✗	✓	✗	✗	✗	✗	✗	✗		0.003
!cap92	854 704.20	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.000	0.003
*cap94	928 941.75	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.000	0.007
#capb	12 979 071.58	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.041	0.560
#capc	11 505 594.33	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.145	0.644
!MO1	1305.95	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.074
!MO1	1305.95	✗	✓	✓	✗	✗	✗	✗	✗	✗		0.059
!MO2	1432.36	✗	✗	✓	✗	✗	✗	✗	✗	✗		0.061
!MO4	1442.24	✓	✓	✗	✗	✗	✗	✗	✗	✗		0.064
!MO5	1408.77	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.002	0.073
#MP1	2686.48	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.002	0.253

#MP3	2623.71	✗	✗	✓	✗	✗	✗	✗	✗	✗		0.297
#MP3	2623.71	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.258
#MP5	2932.33	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.019	0.314
#MP5	2932.33	✗	✓	✓	✗	✗	✗	✗	✗	✗	0.058	0.290
#MQ2	4028.33	✓	✓	✗	✗	✗	✗	✗	✗	✗		0.613
#MQ2	4028.33	✓	✗	✓	✗	✗	✗	✗	✗	✗		0.603
#MQ4	4235.15	✓	✓	✓	✗	✗	✗	✗	✗	✗		0.608
!MR1	2608.15	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.021	1.893
!MR1	2608.15	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.042	1.707
!MR2	2654.73	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.022	1.712
!MR4	2756.04	✓	✗	✓	✗	✗	✗	✗	✗	✗		1.816
!MR5	2505.05	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.038	1.877
!MR5	2505.05	✓	✗	✓	✗	✗	✗	✗	✗	✗	0.015	1.667
#MS1	5283.76	✗	✗	✓	✗	✗	✗	✗	✗	✗		16.405
#MT1	10 069.80	✗	✗	✓	✗	✗	✗	✗	✗	✗	1.032	90.741
#1223UnifS	72 057.00	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.018	0.085
#123UnifS	71 342.00	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.015	0.081
#123UnifS	71 342.00	✓	✓	✓	✗	✗	✗	✗	✗	✗	0.009	0.075
#1323UnifS	72 414.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.016	0.079
#1423UnifS	72 632.00	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.013	0.075
#1623UnifS	74 344.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.005	0.071
#1723UnifS	73 847.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.037	0.096
!1823UnifS	71 903.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.030	0.092
#2023UnifS	73 765.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.008	0.076
#2023UnifS	72 458.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.023	0.086
!2223UnifS	72 409.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.040	0.106
!2223UnifS	71 945.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.029	0.094
!2323UnifS	71 335.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.011	0.078
!2323UnifS	71 335.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.037	0.095
!2523UnifS	73 807.00	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.051	0.113
!2623UnifS	70 929.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.004	0.071
#2823UnifS	76 839.00	✓	✗	✗	✗	✗	✗	✗	✗	✗		0.067
#2823UnifS	75 025.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.008	0.068
#2923UnifS	75 279.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.015	0.077
#323UnifS	73 319.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.026	0.091
!323UnifS	73 319.00	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.007	0.069
!523UnifS	69 251.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.016	0.080
!523UnifS	70 394.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.044	0.099
!723UnifS	73 709.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.050	0.108
#823UnifS	71 860.00	✓	✓	✗	✗	✗	✗	✗	✗	✗	0.013	0.081
#2723UnifS	77 997.00	✗	✓	✓	✗	✗	✗	✗	✗	✗	0.023	0.036
#2823UnifS	75 114.00	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.025	0.038
#2923UnifS	75 577.00	✗	✗	✗	✗	✓	✓	✗	✗	✗	0.015	0.036
#3023UnifS	76 557.00	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.003	0.034
#323UnifS	73 574.00	✓	✗	✗	✗	✗	✗	✗	✓	✓	0.003	0.036
#423UnifS	75 186.00	✓	✓	✓	✗	✓	✓	✓	✓	✓	0.011	0.035
*523UnifS	69 251.00	✗	✗	✗	✗	✗	✗	✓	✓	✓	0.014	0.036

#623UnifS	73 139.00	✗	✓	✓	✗	✗	✗	✓	✓	✗	0.006	0.036
#723UnifS	74 733.00	✓	✓	✓	✓	✓	✓	✓	✓	✓		0.035
#823UnifS	71 949.00	✗	✓	✓	✓	✓	✗	✗	✗	✗	0.003	0.036
#923UnifS	74 237.00	✓	✓	✓	✗	✗	✗	✓	✓	✗	0.002	0.036

6.6. Tabu Search

Tabu Search o Búsqueda Tabú se basa en mantener una lista de elementos tabú que no se pueden modificar durante un número determinado de iteraciones. Así, en el caso de UFLP, Tabu Search utiliza como relación de vecindad las soluciones que tienen una instalación en estado (abierto o cerrado) invertido con respecto a la actual. Elige la que más mejore el valor objetivo actual de todas ellas y añade a la lista tabú la instalación que difiere de la solución actual. Al permanecer esta instalación en la lista tabú se permite al algoritmo averiguar si es positivo que esta instalación se encuentre en el estado propuesto por esta solución, es decir, se encuentran soluciones que mejoran la función objetivo o, por el contrario, debería estar en el estado inverso, pues las soluciones encontradas empeoran la función objetivo. En caso de que las soluciones vecinas no mejorasen la función objetivo, se cambiaría el estado de una instalación de forma aleatoria.

6.6.1. Benchmark

Tabu Search es el mejor algoritmo en relación calidad de las soluciones frente al tiempo de cálculo TS. Alcanza el óptimo en la mayoría de ficheros, no alcanzándolo en 2 partiendo desde ADD y en 4 partiendo desde DROP. Simulated Annealing solo llegaba a una solución subóptima en 2 ficheros desde ADD y en 1 desde DROP. Sin embargo, los tiempos eran mucho mayores, siendo aquí el máximo de menos de 0.2 segundos y en Simulated Annealing de más de 10 segundos. Al igual que con Simulated Annealing, no aparece el problema de suboptimalidad en los problemas UnifS que sí se manifestaba con el resto de metaheurísticas.

Cuadro 6.17: Benchmark Tabu Search ADD

Datos	**BST**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**BST(TS)**	**BST(TE)**
*cap101	796 648.44	✗	✗	✗	✓	✓	✓	✓	✗	0.002	0.007
*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✗	0.000	0.005
*cap103	893 782.11	✗	✓	✗	✓	✓	✗	✓	✗	0.001	0.005
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*cap111	793 439.56	✗	✓	✓	✓	✓	✓	✓	✗	0.003	0.018
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.011
*cap113	893 076.71	✗	✓	✓	✓	✓	✗	✗	✗	0.002	0.019

*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap121	793 439.56	✗	✓	✓	✗	✓	✗	✓	✓	0.000	0.012
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.012
*cap123	893 076.71	✗	✓	✓	✓	✓	✓	✗	✗	0.001	0.015
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap131	793 439.56	✗	✗	✗	✓	✓	✗	✗	✗	0.002	0.017
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✗	0.000	0.017
*cap133	893 076.71	✗	✓	✓	✓	✗	✓	✓	✗	0.002	0.019
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.011
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.001
*cap43	1 010 641.45	✗	✓	✓	✓	✓	✓	✓	✓	0.000	0.001
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap51	1 010 641.45	✗	✓	✓	✓	✓	✓	✓	✓	0.000	0.001
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.001
*cap63	1 010 641.45	✗	✓	✓	✓	✓	✓	✗	✗	0.000	0.003
*cap64	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.001
!cap73	1 001 416.86	✗	✗	✗	✗	✗	✗	✗	✓	0.002	0.002
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap81	796 648.44	✗	✓	✗	✗	✓	✓	✓	✓	0.000	0.003
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✗	✗	0.000	0.006
*cap83	893 782.11	✗	✓	✓	✓	✓	✓	✓	✗	0.001	0.006
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*cap91	796 648.44	✗	✗	✗	✗	✓	✓	✗	✗	0.000	0.006
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✗	✗	0.001	0.007
*cap93	893 782.11	✗	✓	✓	✓	✓	✓	✓	✗	0.002	0.006
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	0.001	0.421
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✓	✗	0.030	0.549
*capc	11 505 594.33	✗	✓	✓	✓	✓	✓	✓	✗	0.155	0.568
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✗	0.001	0.074
*MO2	1432.36	✗	✓	✓	✓	✓	✓	✓	✓	0.001	0.062
*MO3	1516.77	✗	✓	✗	✓	✓	✓	✓	✗	0.005	0.072
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.060
*MO5	1408.77	✗	✓	✓	✓	✓	✓	✓	✓	0.001	0.061
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✗	0.009	0.307
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓	0.007	0.297
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✗	0.005	0.283
*MP4	2938.75	✗	✓	✓	✓	✓	✓	✓	✗	0.033	0.307
*MP5	2932.33	✓	✓	✓	✓	✓	✓	✓	✓	0.002	0.258
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓	0.007	0.611
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓	0.007	0.597
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓	0.003	0.614
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓	0.004	0.606

*MQ5	4080.74	✗	✓	✓	✓	✓	✓	✓	✓	0.103	0.754
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	0.012	1.670
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	0.032	1.881
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	0.037	1.928
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓	0.008	1.664
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	0.008	1.660
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓	0.120	16.562
!MT1	10 069.80	✗	✓	✓	✓	✓	✓	✓	✓	15.748	109.504
*1023UnifS	71 840.00	✗	✓	✓	✓	✓	✓	✗	✗	0.026	0.090
*1123UnifS	75 063.00	✗	✓	✓	✓	✓	✗	✗	✗	0.014	0.080
*1223UnifS	71 756.00	✗	✗	✗	✗	✗	✓	✗	✗	0.032	0.091
*123UnifS	71 342.00	✗	✓	✓	✓	✓	✓	✓	✓	0.004	0.063
*1323UnifS	72 414.00	✗	✗	✗	✗	✓	✓	✗	✗	0.035	0.096
*1423UnifS	72 632.00	✗	✗	✗	✓	✗	✓	✗	✗	0.018	0.083
!1523UnifS	77 566.00	✗	✓	✓	✓	✓	✓	✗	✗	0.012	0.074
*1623UnifS	74 344.00	✗	✓	✓	✗	✓	✓	✗	✗	0.016	0.074
*1723UnifS	73 847.00	✗	✓	✓	✓	✓	✗	✗	✗	0.012	0.075
*1823UnifS	71 903.00	✗	✓	✓	✓	✓	✓	✗	✗	0.004	0.063
#1923UnifS	75 581.00	✗	✓	✗	✗	✗	✗	✗	✗	0.051	0.117
*2023UnifS	72 458.00	✗	✓	✓	✓	✓	✓	✗	✗	0.023	0.085
*2123UnifS	72 631.00	✗	✗	✗	✗	✗	✗	✓	✗	0.057	0.112
#2223UnifS	72 068.00	✗	✗	✗	✗	✗	✗	✓	✗	0.025	0.080
*223UnifS	74 148.00	✗	✓	✓	✗	✓	✓	✓	✗	0.041	0.107
*2323UnifS	71 335.00	✗	✓	✓	✓	✓	✗	✗	✗	0.016	0.082
*2423UnifS	75 342.00	✗	✗	✗	✓	✗	✓	✗	✗	0.053	0.117
#2523UnifS	73 926.00	✗	✗	✗	✓	✓	✗	✓	✗	0.015	0.080
*2623UnifS	70 929.00	✗	✓	✓	✓	✓	✓	✗	✗	0.003	0.062
*2723UnifS	77 314.00	✓	✓	✓	✓	✓	✓	✗	✗	0.003	0.069
*2823UnifS	75 025.00	✗	✗	✗	✗	✗	✓	✗	✗	0.053	0.112
*2923UnifS	75 279.00	✗	✓	✓	✓	✓	✓	✓	✗	0.006	0.072
#3023UnifS	75 845.00	✓	✓	✓	✓	✓	✓	✓	✗	0.004	0.068
*323UnifS	73 319.00	✗	✗	✓	✓	✓	✓	✗	✗	0.017	0.078
*423UnifS	74 979.00	✗	✗	✓	✗	✓	✗	✓	✗	0.007	0.075
*523UnifS	69 251.00	✗	✓	✗	✓	✓	✓	✗	✗	0.031	0.096
*623UnifS	72 992.00	✗	✗	✗	✗	✓	✓	✓	✗	0.038	0.096
*723UnifS	73 709.00	✗	✓	✓	✓	✓	✓	✗	✗	0.030	0.091
*823UnifS	71 860.00	✓	✓	✓	✓	✓	✓	✗	✗	0.011	0.077
*923UnifS	74 100.00	✗	✓	✓	✗	✓	✗	✓	✗	0.022	0.084

Cuadro 6.18: Benchmark Tabu Search DROP

Datos	**BST**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**BST(TS)**	**BST(TE)**
*cap101	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.004

*cap102	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.003
*cap103	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*cap104	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.003
*cap111	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.014
*cap112	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.013
*cap113	893 076.71	✗	✓	✓	✓	✓	✓	✓	✗	0.001	0.017
*cap114	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.014
*cap121	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.014
*cap122	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.013
*cap123	893 076.71	✗	✓	✓	✓	✓	✓	✓	✗	0.000	0.015
*cap124	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.013
*cap131	793 439.56	✓	✓	✓	✓	✓	✓	✓	✓		0.012
*cap132	851 495.33	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.013
*cap133	893 076.71	✗	✓	✓	✓	✓	✓	✓	✓	0.000	0.012
*cap134	928 941.75	✓	✓	✓	✓	✓	✓	✗	✓	0.000	0.017
*cap41	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap42	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap43	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap44	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✗	0.000	0.002
*cap51	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap61	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap62	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap63	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.001
!cap64	1 023 916.86	✗	✗	✗	✗	✗	✗	✗	✓	0.001	0.001
*cap71	932 615.75	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap72	977 799.40	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap73	1 010 641.45	✓	✓	✓	✓	✓	✓	✓	✓		0.001
*cap74	1 034 976.98	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.001
*cap81	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*cap82	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.004
*cap83	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*cap84	928 941.75	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.003
*cap91	796 648.44	✓	✓	✓	✓	✓	✓	✓	✓		0.003
*cap92	854 704.20	✓	✓	✓	✓	✓	✓	✓	✓	0.000	0.003
*cap93	893 782.11	✓	✓	✓	✓	✓	✓	✓	✓		0.004
*cap94	928 941.75	✓	✓	✓	✓	✓	✓	✓	✗	0.000	0.007
*capa	17 156 454.48	✓	✓	✓	✓	✓	✓	✓	✓	0.013	0.497
*capb	12 979 071.58	✓	✓	✓	✓	✓	✓	✗	✗	0.041	0.550
*capc	11 505 594.33	✗	✓	✓	✓	✓	✓	✗	✗	0.145	0.623
*MO1	1305.95	✓	✓	✓	✓	✓	✓	✓	✓		0.059
*MO2	1432.36	✓	✓	✓	✓	✓	✓	✓	✓		0.061
*MO3	1516.77	✗	✗	✗	✓	✓	✓	✓	✗	0.008	0.072
*MO4	1442.24	✓	✓	✓	✓	✓	✓	✓	✓		0.059
*MO5	1408.77	✗	✓	✓	✓	✓	✓	✗	✗	0.002	0.073
*MP1	2686.48	✓	✓	✓	✓	✓	✓	✓	✓	0.002	0.253
*MP2	2904.86	✓	✓	✓	✓	✓	✓	✓	✓		0.259
*MP3	2623.71	✓	✓	✓	✓	✓	✓	✓	✓		0.258

*MP4	2938.75	✗	✓	✓	✓	✓	✓	✓	✗	0.035	0.311
*MP5	2932.33	✗	✓	✓	✓	✓	✓	✓	✓	0.019	0.290
*MQ1	4091.01	✓	✓	✓	✓	✓	✓	✓	✓		0.605
*MQ2	4028.33	✓	✓	✓	✓	✓	✓	✓	✓		0.603
*MQ3	4275.43	✓	✓	✓	✓	✓	✓	✓	✓		0.614
*MQ4	4235.15	✓	✓	✓	✓	✓	✓	✓	✓		0.608
*MQ5	4080.74	✓	✓	✓	✓	✓	✓	✓	✓	0.020	0.695
*MR1	2608.15	✓	✓	✓	✓	✓	✓	✓	✓	0.021	1.707
*MR2	2654.73	✓	✓	✓	✓	✓	✓	✓	✓	0.022	1.712
*MR3	2788.25	✓	✓	✓	✓	✓	✓	✓	✓	0.046	1.852
*MR4	2756.04	✓	✓	✓	✓	✓	✓	✓	✓		1.661
*MR5	2505.05	✓	✓	✓	✓	✓	✓	✓	✓	0.015	1.667
*MS1	5283.76	✓	✓	✓	✓	✓	✓	✓	✓		16.405
!MT1	10 069.80	✗	✓	✓	✓	✓	✓	✓	✓	1.032	90.741
*1023UnifS	71 840.00	✗	✗	✓	✓	✓	✓	✓	✗	0.019	0.078
*1123UnifS	75 063.00	✗	✓	✗	✗	✓	✓	✓	✗	0.024	0.089
#1223UnifS	72 052.00	✗	✓	✗	✗	✗	✗	✗	✗	0.047	0.113
*123UnifS	71 342.00	✗	✓	✓	✓	✓	✓	✓	✓	0.009	0.075
*1323UnifS	72 414.00	✗	✓	✓	✓	✓	✗	✗	✗	0.016	0.079
*1423UnifS	72 632.00	✗	✓	✗	✓	✓	✗	✗	✗	0.013	0.075
!1523UnifS	77 566.00	✗	✗	✗	✗	✓	✗	✗	✗	0.015	0.077
*1623UnifS	74 344.00	✗	✓	✓	✓	✓	✓	✗	✓	0.005	0.065
*1723UnifS	73 847.00	✗	✓	✓	✓	✓	✓	✗	✗	0.037	0.096
*1823UnifS	71 903.00	✗	✓	✗	✓	✓	✓	✓	✗	0.030	0.092
#1923UnifS	75 581.00	✗	✗	✗	✓	✗	✓	✗	✗	0.053	0.111
*2023UnifS	72 458.00	✗	✗	✓	✓	✓	✓	✗	✗	0.023	0.086
*2123UnifS	72 631.00	✗	✓	✓	✓	✓	✗	✓	✗	0.025	0.088
*2223UnifS	71 945.00	✗	✗	✗	✓	✓	✓	✗	✗	0.029	0.094
*223UnifS	74 148.00	✗	✗	✓	✓	✓	✓	✓	✓	0.010	0.068
*2323UnifS	71 335.00	✗	✓	✓	✓	✓	✓	✗	✗	0.011	0.078
*2423UnifS	75 342.00	✗	✓	✓	✓	✗	✓	✗	✗	0.034	0.101
*2523UnifS	73 807.00	✗	✗	✗	✗	✓	✗	✗	✗	0.051	0.113
*2623UnifS	70 929.00	✗	✓	✓	✓	✓	✓	✓	✗	0.004	0.071
*2723UnifS	77 314.00	✗	✓	✓	✓	✓	✓	✗	✗	0.004	0.068
*2823UnifS	75 025.00	✗	✗	✗	✗	✓	✓	✗	✗	0.008	0.068
*2923UnifS	75 279.00	✗	✓	✓	✓	✓	✓	✗	✗	0.015	0.077
*3023UnifS	75 654.00	✗	✗	✗	✓	✗	✗	✗	✗	0.067	0.132
*323UnifS	73 319.00	✗	✗	✗	✓	✓	✗	✗	✗	0.007	0.069
*423UnifS	74 979.00	✗	✗	✗	✓	✗	✗	✗	✗	0.105	0.170
*523UnifS	69 251.00	✗	✓	✓	✓	✓	✓	✗	✗	0.016	0.080
*623UnifS	72 992.00	✗	✓	✓	✓	✓	✓	✓	✗	0.010	0.066
*723UnifS	73 709.00	✗	✓	✓	✓	✗	✓	✗	✗	0.050	0.108
*823UnifS	71 860.00	✗	✓	✓	✗	✓	✓	✗	✗	0.013	0.081
*923UnifS	74 100.00	✗	✓	✓	✗	✓	✓	✗	✗	0.009	0.076

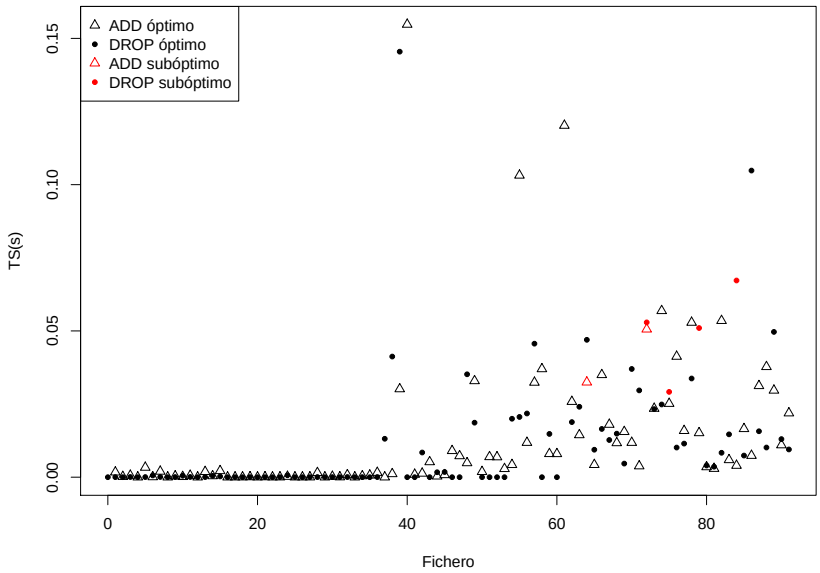

Figura 6.8: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Tabu Search.

6.7. Algoritmo Genético

Los algoritmos genéticos, al igual que el resto de metaheurísticas, permiten explorar espacios de soluciones inabarcables por un método exacto debido a la explosión combinatoria derivada del tamaño del problema. Han experimentado un auge gracias a las redes neuronales, ya que los algoritmos genéticos son uno de los métodos más utilizados para entrenarlas. Dadas sus características, también cumple las características de una metaheurística, por lo que permite resolver el problema UFLP.

Este tipo de algoritmos tienen un enfoque biológico que trata de replicar el mecanismo de selección natural que lleva a la evolución de una población, de modo que a partir de cualquier solución esta pueda evolucionar hasta aproximarse al óptimo global. El término técnico para las soluciones en los algoritmos genéticos es el de cromosoma, que suelen representarse mediante un array de bits. Tal y como ocurre en la naturaleza, el algoritmo genético consta de varios estadios cíclicos: cruce de individuos, mutación de individuos y selección elitista (figura).

En el cruce de individuos los cromosomas se cruzan a pares bajo una probabilidad dada. Durante este cruce los cromosomas padre se dividen por un punto de corte elegido de forma aleatoria. A la parte más a la izquierda del corte de uno de los cromosomas se le adhiere la parte más a la derecha del otro cromosoma padre, y viceversa. Esto se denomina *single point crossover*. Existen cruces con más de un punto de corte y diferentes formas de escoger este punto de corte, aunque en este trabajo solo se explora esta posibilidad.

La mutación de individuos es un mecanismo de diversificación de las soluciones, al igual que el cruce. Esta ocurre con una cierta probabilidad, que, al igual que en la naturaleza, en general será baja. Este mecanismo consiste en invertir de forma aleatoria el estado de uno de los elementos del cromosma (abrir o cerrar una instalación). Los individuos mutados se añaden a un *pool* complementario.

La selección elitista es, en cambio, un mecanismo de intensificación. Consiste en seleccionar los mejores cromosomas de todos los disponibles (incluidos los del *pool* de mutantes) hasta alcanzar el número fijo de cromosomas que debe tener la población (parámetro del algoritmo). El resto se desechan, pues no cumplen las condiciones para sobrevivir (empeoran la función objetivo en exceso).

En este trabajo se ha implementado el algoritmo genético desarrollado por Tohyama, Ida y Matsueda (Tohyama, Ida y Matsueda, 2011).

6.7.1. Benchmark

Los resultados obtenidos con el algoritmo genético empleado en este trabajo no han sido buenos. Respaldan los presentados en el artículo original en cuanto que se llega al óptimo con los ficheros cap. Sin embargo, dicho artículo también sostiene que se alcanza para las instancias MO*, MP* y MQ*. No se pronuncia sobre los ficheros UnifS, para los que se comprueba en este trabajo que el algoritmo nunca llega al óptimo. Estos ficheros son más

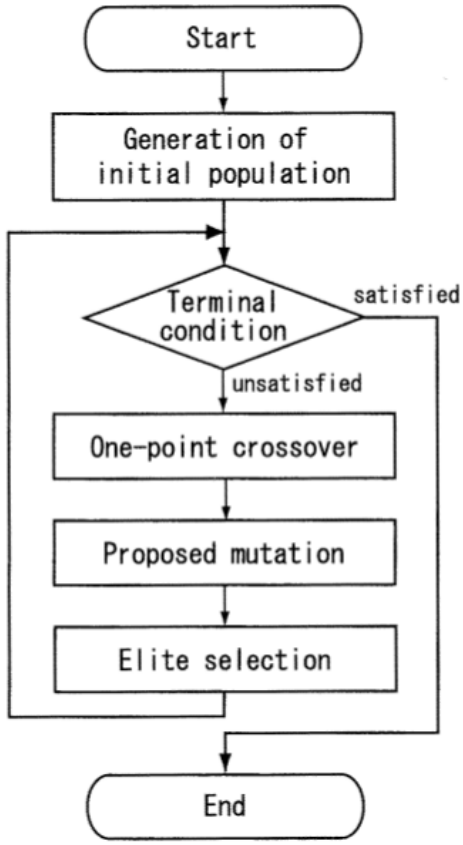

Figura 6.9: Flujo general de un algoritmo genético (Tohyama, Ida y Matsueda, 2011).

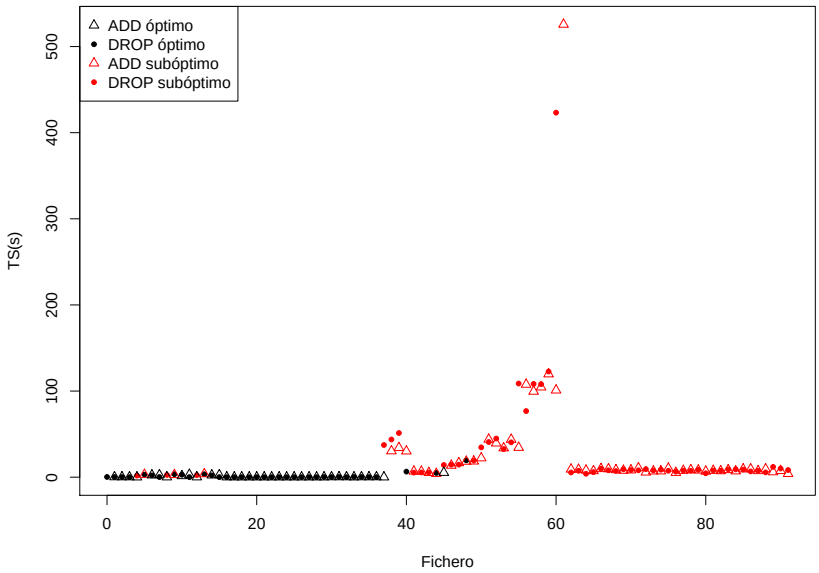

Figura 6.10: Comparación de las soluciones obtenidas partiendo de la solución base de la heurística ADD y DROP con Algoritmo Genético.

sencillos que las instancais M* para las que sí se alcanzó el óptimo, por lo que probablemente estos malos resultados se deban a una mala elección de parámetros. En concreto, se ha utilizado el conjunto de parámetros recomendado en el artículo. Sin embargo, estos podrían no ser adecuados para los ficheros UnifS. Dado que no se han obtenido los óptimos para todas las instancias MO*, MP* y MQ* esto induce a pensar que los parámetros tampoco son los adecuados para estos ficheros.

El comportamiento que aquí se observa es, por otro lado, esperable en un algoritmo genético, que necesita un ajuste perfecto de sus parámetros para funcionar correctamente. La solución para conseguir un mayor índice de optimalidad pasaría por encontrar estos parámetros, pero dado el elevado tiempo que toman las ejecuciones del algoritmo y el número de instancias tratadas esto sobrepasa los límites del objetivo del trabajo.

Cuadro 6.19: Benchmark Genetic Algorithm ADD

Datos	**BEST**	**AVG**	**WS**	μ**(TS)**	μ**(TE)**
cap101.txt	796 648.438	796 648.438	796 648.438	0.190	4.091
cap102.txt	854 704.200	854 704.200	854 704.200	0.154	3.730
cap103.txt	893 782.113	893 782.113	893 782.113	0.157	3.542
cap104.txt	928 941.750	928 941.750	928 941.750	0.020	3.445
cap111.txt	794 299.850	796 132.671	797 639.050	2.982	5.866
cap112.txt	851 495.325	852 298.308	853 346.325	2.339	5.630
cap113.txt	893 076.713	893 617.816	894 008.138	2.586	5.449
cap114.txt	928 941.750	928 941.750	928 941.750	0.073	5.342
cap121.txt	794 448.400	796 288.923	798 134.375	2.762	5.848
cap122.txt	851 495.325	852 465.259	852 864.738	1.495	5.621
cap123.txt	893 076.713	893 177.299	893 732.975	3.037	5.450
cap124.txt	928 941.750	928 941.750	928 941.750	0.092	5.347
cap131.txt	794 299.850	796 265.876	797 441.900	3.780	5.910
cap132.txt	851 495.325	852 439.970	853 782.813	2.327	5.717
cap133.txt	893 076.713	893 265.319	893 907.775	2.589	5.455
cap134.txt	928 941.750	928 941.750	928 941.750	0.074	5.330
cap41.txt	932 615.750	932 615.750	932 615.750	0.017	3.262
cap42.txt	977 799.400	977 799.400	977 799.400	0.021	2.823
cap43.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.011	2.572
cap44.txt	1 034 976.975	1 034 976.975	1 034 976.975	0.008	2.453
cap51.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.010	2.570
cap61.txt	932 615.750	932 615.750	932 615.750	0.015	3.270
cap62.txt	977 799.400	977 799.400	977 799.400	0.022	2.818
cap63.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.015	2.574
cap64.txt	1 034 976.975	1 034 976.975	1 034 976.975	0.006	2.454
cap71.txt	932 615.750	932 615.750	932 615.750	0.017	3.259
cap72.txt	977 799.400	977 799.400	977 799.400	0.020	2.820
cap73.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.015	2.573
cap74.txt	1 034 976.975	1 034 976.975	1 034 976.975	0.010	2.450
cap81.txt	796 648.438	796 648.438	796 648.438	0.213	4.056
cap82.txt	854 704.200	854 704.200	854 704.200	0.192	3.728

cap83.txt	893 782.113	893 782.113	893 782.113	0.175	3.528
cap84.txt	928 941.750	928 941.750	928 941.750	0.022	3.421
cap91.txt	796 648.438	796 648.438	796 648.438	0.186	4.058
cap92.txt	854 704.200	854 704.200	854 704.200	0.138	3.735
cap93.txt	893 782.113	893 782.113	893 782.113	0.203	3.538
cap94.txt	928 941.750	928 941.750	928 941.750	0.025	3.434
capa.txt	17 346 752.156	17 737 646.822	17 892 447.238	30.335	76.122
capb.txt	13 151 458.879	13 263 112.710	13 340 697.582	34.157	82.856
capc.txt	11 833 126.764	12 017 340.892	12 125 682.034	30.164	86.956
MO1	1305.951	1320.528	1328.259	6.764	11.636
MO2	1445.705	1459.635	1466.440	6.669	11.664
MO3	1533.096	1540.650	1551.653	5.186	11.560
MO4	1463.657	1476.113	1496.468	4.076	11.709
MO5	1408.766	1410.963	1416.103	5.294	11.578
MP1	2702.363	2758.832	2791.900	13.601	37.237
MP2	2958.545	2984.576	3009.399	16.534	37.155
MP3	2686.303	2747.896	2776.321	18.214	37.098
MP4	2938.750	3007.523	3034.380	18.473	37.092
MP5	2945.699	2987.277	3006.856	22.164	37.096
MQ1	4213.596	4271.307	4331.105	43.860	78.641
MQ2	4140.187	4226.541	4260.921	39.410	78.874
MQ3	4450.598	4502.964	4538.352	33.750	78.261
MQ4	4332.461	4396.479	4443.723	43.712	78.551
MQ5	4186.705	4242.075	4268.377	34.389	78.717
MR1	2792.037	2847.782	2885.135	107.633	212.204
MR2	2822.691	2877.149	2916.153	99.536	212.500
MR3	2921.942	2944.326	2971.417	104.455	212.313
MR4	2923.469	2966.947	3021.840	119.686	212.873
MR5	2678.307	2725.949	2773.661	100.939	213.019
MS1	5702.247	5832.311	5923.596	525.649	917.594
MT1	10 918.160	11 062.612	11 279.975	1856.021	4221.555
1023UnifS.txt	77 342.000	79 150.900	80 062.000	9.091	16.212
1123UnifS.txt	79 542.000	81 300.500	82 421.000	8.863	15.910
1223UnifS.txt	76 210.000	77 676.000	78 870.000	7.903	15.833
123UnifS.txt	74 579.000	76 873.300	77 979.000	7.249	15.947
1323UnifS.txt	74 338.000	76 267.100	77 280.000	9.694	15.920
1423UnifS.txt	78 670.000	80 051.100	80 973.000	9.528	15.885
1523UnifS.txt	82 392.000	82 998.400	84 196.000	8.453	16.172
1623UnifS.txt	79 435.000	81 017.500	82 387.000	7.730	16.085
1723UnifS.txt	76 688.000	78 350.500	79 655.000	8.125	16.300
1823UnifS.txt	75 875.000	77 298.900	78 454.000	10.618	16.195
1923UnifS.txt	78 266.000	79 831.900	81 493.000	5.667	16.326
2023UnifS.txt	77 108.000	80 521.000	81 995.000	6.909	15.981
2123UnifS.txt	78 825.000	79 821.000	80 982.000	7.031	15.872
2223UnifS.txt	76 281.000	78 266.900	80 030.000	10.245	15.976
223UnifS.txt	77 404.000	79 183.600	80 707.000	5.273	16.068
2323UnifS.txt	76 651.000	78 532.000	80 143.000	7.633	15.804

2423UnifS.txt	82 341.000	83 330.200	84 646.000	8.447	16.039
2523UnifS.txt	77 206.000	79 662.100	81 192.000	7.984	16.044
2623UnifS.txt	77 017.000	78 299.500	79 509.000	6.739	16.015
2723UnifS.txt	80 568.000	82 346.400	83 680.000	7.620	15.727
2823UnifS.txt	80 632.000	81 784.900	83 222.000	7.224	15.944
2923UnifS.txt	80 203.000	81 277.500	82 079.000	8.512	16.383
3023UnifS.txt	79 723.000	81 566.900	83 651.000	7.063	16.013
323UnifS.txt	78 332.000	79 790.900	81 638.000	9.382	16.157
423UnifS.txt	80 411.000	81 292.500	82 134.000	9.406	16.200
523UnifS.txt	75 289.000	76 329.500	77 077.000	7.351	15.905
623UnifS.txt	77 991.000	79 131.200	80 557.000	9.415	16.119
723UnifS.txt	77 417.000	79 718.500	80 981.000	5.965	16.272
823UnifS.txt	75 928.000	76 579.200	77 374.000	7.480	16.137
923UnifS.txt	78 720.000	79 424.800	80 579.000	4.039	16.323

Cuadro 6.20: Benchmark Genetic Algorithm DROP

Datos	**BEST**	**AVG**	**WS**	μ**(TS)**	μ**(TE)**
data/cap101.txt	796 648.438	796 648.438	796 648.438	0.269	
data/cap102.txt	854 704.200	854 704.200	854 704.200	0.153	
data/cap103.txt	893 782.113	893 782.113	893 782.113	0.110	
data/cap104.txt	928 941.750	928 941.750	928 941.750	0.026	
data/cap111.txt	795 708.950	797 169.741	798 357.350	1.845	
data/cap112.txt	851 495.325	852 487.468	854 132.288	2.943	
data/cap113.txt	893 076.713	893 557.724	894 008.138	2.115	
data/cap114.txt	928 941.750	928 941.750	928 941.750	0.149	
data/cap121.txt	794 448.400	796 462.811	798 390.975	2.665	
data/cap122.txt	851 495.325	852 438.550	853 039.538	3.510	
data/cap123.txt	893 076.713	893 658.101	894 008.138	2.796	
data/cap124.txt	928 941.750	928 941.750	928 941.750	0.090	
data/cap131.txt	794 299.850	796 166.630	797 676.300	2.432	
data/cap132.txt	851 495.325	852 510.058	853 171.525	3.425	
data/cap133.txt	893 076.713	893 305.813	893 782.113	2.092	
data/cap134.txt	928 941.750	928 941.750	928 941.750	0.077	
data/cap41.txt	932 615.750	932 615.750	932 615.750	0.019	
data/cap42.txt	977 799.400	977 799.400	977 799.400	0.018	
data/cap43.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.015	
data/cap44.txt	1 034 976.975	1 034 976.975	1 034 976.975	0.008	
data/cap51.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.013	
data/cap61.txt	932 615.750	932 615.750	932 615.750	0.019	
data/cap62.txt	977 799.400	977 799.400	977 799.400	0.025	
data/cap63.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.012	
data/cap64.txt	1 034 976.975	1 034 976.975	1 034 976.975	0.007	
data/cap71.txt	932 615.750	932 615.750	932 615.750	0.022	
data/cap72.txt	977 799.400	977 799.400	977 799.400	0.029	
data/cap73.txt	1 010 641.450	1 010 641.450	1 010 641.450	0.014	

data/cap74.txt	1 034 976.975	1 034 976.975	1 034 976.975	0.010
data/cap81.txt	796 648.438	796 648.438	796 648.438	0.134
data/cap82.txt	854 704.200	854 704.200	854 704.200	0.165
data/cap83.txt	893 782.113	893 782.113	893 782.113	0.169
data/cap84.txt	928 941.750	928 941.750	928 941.750	0.023
data/cap91.txt	796 648.438	796 648.438	796 648.438	0.178
data/cap92.txt	854 704.200	854 704.200	854 704.200	0.178
data/cap93.txt	893 782.113	893 782.113	893 782.113	0.099
data/cap94.txt	928 941.750	928 941.750	928 941.750	0.015
data/capa.txt	17 639 346.016	17 818 654.727	18 055 665.297	37.372
data/capb.txt	13 118 685.212	13 334 529.729	13 454 527.834	43.799
data/capc.txt	11 893 850.924	12 013 353.964	12 131 751.067	51.283
data/M/O/MO1	1317.893	1321.839	1330.469	6.501
data/M/O/MO2	1451.053	1458.963	1466.440	5.337
data/M/O/MO3	1533.096	1539.327	1544.630	5.492
data/M/O/MO4	1468.996	1480.387	1496.792	5.485
data/M/O/MO5	1408.766	1413.013	1423.934	4.936
data/M/P/MP1	2727.390	2758.686	2807.335	14.183
data/M/P/MP2	2979.971	3003.921	3023.147	14.531
data/M/P/MP3	2689.959	2747.846	2796.917	14.580
data/M/P/MP4	2975.995	3013.001	3038.681	19.520
data/M/P/MP5	2958.095	2997.422	3025.380	19.277
data/M/Q/MQ1	4245.123	4286.691	4324.613	34.625
data/M/Q/MQ2	4166.254	4234.615	4281.378	40.825
data/M/Q/MQ3	4473.397	4498.565	4528.809	44.898
data/M/Q/MQ4	4299.701	4386.375	4443.723	32.624
data/M/Q/MQ5	4159.781	4235.853	4292.259	40.505
data/M/R/MR1	2785.481	2844.022	2903.042	108.765
data/M/R/MR2	2824.754	2872.387	2906.880	76.766
data/M/R/MR3	2894.163	2957.324	2999.961	108.319
data/M/R/MR4	2930.396	2982.039	3024.584	108.027
data/M/R/MR5	2680.584	2739.564	2780.558	122.687
data/M/S/MS1	5673.305	5829.847	5931.598	423.270
data/M/T/MT1	11 046.893	11 174.145	11 332.723	2331.361
data/1023UnifS.txt	78 182.000	79 067.300	79 838.000	5.482
data/1123UnifS.txt	80 037.000	81 726.500	83 576.000	7.438
data/1223UnifS.txt	76 407.000	77 431.100	78 507.000	4.004
data/123UnifS.txt	75 454.000	76 571.500	77 834.000	5.912
data/1323UnifS.txt	74 598.000	76 278.100	76 997.000	9.794
data/1423UnifS.txt	77 808.000	79 785.600	80 627.000	7.942
data/1523UnifS.txt	81 435.000	82 948.200	83 990.000	7.140
data/1623UnifS.txt	78 978.000	81 317.500	82 409.000	9.128
data/1723UnifS.txt	76 485.000	78 156.300	79 567.000	8.240
data/1823UnifS.txt	76 431.000	77 527.400	78 852.000	7.982
data/1923UnifS.txt	79 007.000	79 951.500	81 018.000	9.313
data/2023UnifS.txt	75 051.000	80 971.400	82 992.000	7.799
data/2123UnifS.txt	78 187.000	79 273.400	80 834.000	8.662

data/2223UnifS.txt	76 488.000	78 017.300	79 475.000	7.469
data/223UnifS.txt	77 380.000	79 082.600	80 492.000	6.471
data/2323UnifS.txt	77 196.000	78 698.600	79 700.000	7.732
data/2423UnifS.txt	82 376.000	83 186.100	84 274.000	7.349
data/2523UnifS.txt	77 988.000	79 407.000	80 335.000	8.480
data/2623UnifS.txt	77 168.000	78 522.700	79 990.000	4.554
data/2723UnifS.txt	82 398.000	83 100.900	83 990.000	7.828
data/2823UnifS.txt	80 917.000	82 084.000	83 142.000	7.438
data/2923UnifS.txt	80 009.000	81 138.700	82 527.000	9.071
data/3023UnifS.txt	80 522.000	81 486.600	82 236.000	9.078
data/323UnifS.txt	77 524.000	79 639.600	81 671.000	8.843
data/423UnifS.txt	80 235.000	81 681.700	82 369.000	6.699
data/523UnifS.txt	74 688.000	76 389.100	77 986.000	7.460
data/623UnifS.txt	75 902.000	78 638.900	80 374.000	5.566
data/723UnifS.txt	78 223.000	79 321.400	80 154.000	11.973
data/823UnifS.txt	76 441.000	77 094.100	78 184.000	10.060
data/923UnifS.txt	79 294.000	79 928.900	80 986.000	8.298

Capítulo 7

Conclusiones

En este trabajo se ha estudiado el problema UFLP y diferentes formas de resolverlo, con enfoque exacto y aproximado. Se ha comprobado que los metodos exactos ofrecen buenos resultados para ficheros pequeños, especialmente el algoritmo *Branch and Bound* con la formulación fuerte. Sin embargo, no son capaces de alcanzar el óptimo en el tiempo límite de 60 segundos que se ha establecido para la experimentación. Los métodos aproximados, sin embargo, superan ampliamente a los métodos exactos tanto en tiempo como en la calidad de las soluciones obtenidas. Ni siquiera los tiempos de los ficheros pequeños son tan buenos con la formulación fuerte como con las metaheurísticas. Esto queda patente en la comparación de la figura 3.4 con cualquiera de las figuras de tiempos de solución de las metaheurísticas, e.g. la figura 6.2. Si bien es cierto que estas últimas no incluyen los tiempos de obtención de la solución greedy, en la tabla 4.1 se puede ver como para los ficheros UnifS y cap los tiempos son muy inferiores a 1s. Además, los métodos aproximados no alcanzaron un *índice de optimalidad* de 1 para los ficheros UnifS ni cap, lo cual sí ocurre en la mayoría de metaheurísticas. Otra ventaja que tienen las metaheurísticas sobre los métodos exactos es que, a menos que se esté dispuesto a asumir elevados costes de desarrollo asociados a la implementación de estos métodos, requieren de un *solver* específico para Programación Entera. Este tipo de software tiene un precio elevado, aunque a veces existan versiones de prueba o licencias para estudiantes con limitaciones. Estas limitaciones suelen estar en las dimensiones máximas de los problemas a resolver, por lo que tan solo se admitirían los problemas más pequeños. Las metaheurísticas, por otro lado, se caracterizan por ser métodos efectivos (tal y como demuestran los resultados) y sin coste asociado, lo que permite una implementación gratuita por parte del investigador o la empresa interesada en resolver una instancia concreta de UFLP. Por estos motivos, la primera conclusión del trabajo es que los métodos exactos no son adecuados para instancias grandes de UFLP, lo que en términos prácticos significa que no deberían usarse en contextos realistas, pues los problemas M* son lo más parecido a este tipo de casos, en los que las metaheurísticas han demostrado su superioridad.

Las metaheurísticas estudiadas presentan diferencias de rendimiento entre ellas. Antes de empezar el análisis conjunto cabe destacar que esto puede venir condicionado por la elección de parámetros, que tan frecuentemente es calificado como un arte. La dificultad de obtener

los parámetros más adecuados radica en que no solo son específicos de cada problema, sino que, en muchos casos, también lo son de la instancia. Es muy probable que debido a esto, la metaheurística Late Acceptance no haya presentado un *índice de optimalidad* especialmente alto, cuando es uno de los algotimos más premiados de los estudiados (Burke y Bykov, 2012).

Dos de las metaheurísticas estudiadas han destacado sobre las demás en *índice de optimalidad* según la figura 7.2: Simulated Annealing y Tabu Search. La primera ha alcanzado el óptimo en el 98.91 % de los ficheros partiendo de la solución de ADD y el 97.82 % desde DROP, mientas que la segunda llegó al óptimo en el 95.6 % de los ficheros desde ADD y en el 97.82 % desde DROP. Aunque Simulated Annealing es ligeramente superior a la Búsqueda Tabú según este criterio, si se atiende a la figure 7.1, que mide el tiempo de solución del fichero MT1 (dado que es el fichero más difícil de los estudiados en este trabajo, se ha considerado que es, también, el de más interés) la Búsqueda Tabú es mucho mejor. Ambas metaheurísticas alcanzan el óptimo (aunque no es el óptimo teórico suministrado con la documentación del fichero ya se ha alcanzado en otros trabajos, como ya se ha especificado previamente), pero la Búsqueda Tabú llega en 1.032s partiendo de DROP, frente a los 87.56s de Simulated Annealing. Consigue, incluso, el mínimo tiempo de todos los algoritmos probados. Por tanto, se recomienda su uso cuando se trate de resolver el problema UFLP. El Algoritmo Genético no aparece representado en la figura 7.1, dado su elevado tiempo de solución, que desvirtuaría la gráfica (además, no alcanzó el óptimo).

El resto de metaheurísticas obtienen unos resultados más discretos. Threshold Accepting se acerca al 80 % de ficheros en los que alcanzó el óptimo partiendo de DROP y resulta llamativo que Demon Algorithm se encuentre en tercer lugar, partiendo de la misma heurística de construcción, con un 71.74 % (recordar que se ha implementado la versión original de Creutz, que no estaba pensada para problemas de optimización). Tanto Late Acceptance, como las dos versiones de Old Bachelor llegan a unos resultados muy similares, tanto a nivel de *índice de optimalidad* como de tiempo de solución. El algoritmo genético obtiene resultados muy malos (40.21 % partiendo de ADD y %38.04 partiendo de DROP). Esto se debe a un mal ajuste de parámetros (consultar la sección correspondiente del capítulo 6 para más información).

Por último, las heurísticas de construcción parecen tener relevancia en los resultados finales, especialmente entre las metaheurísticas con peor rendimiento. Así, puede verse una tendencia clara de mayor *índice de optimalidad* partiendo de la solución de DROP, según la figura 7.1 (no debe tenerse en cuenta el Algoritmo Genético a tales efectos, pues la solución de la que se parte no parece tener relevancia, dado que la aleatorización y la recombinación diluyen su contribución). Esta tendencia tiene como excepción Simulated Annealing, pero la diferencia partiendo de las dos heurísticas es tan pequeña que puede deberse a la aleatoriedad del método. Las implicaciones en el tiempo de solución no son tan evidentes, parece ser dependiente de la metaheurística. Los resultados aquí presentados constituyen una guía de elección de heurísticas de construcción en función de la metaheurística empleada.

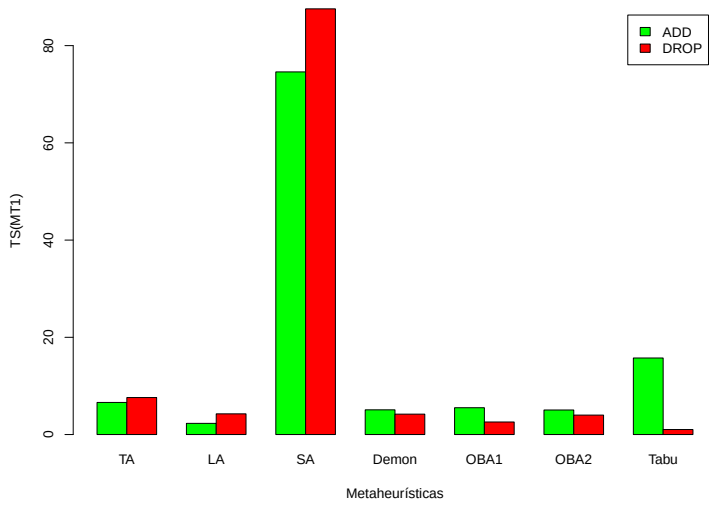

Figura 7.1: Comparativa TS fichero MT1 entre metaheurísticas.

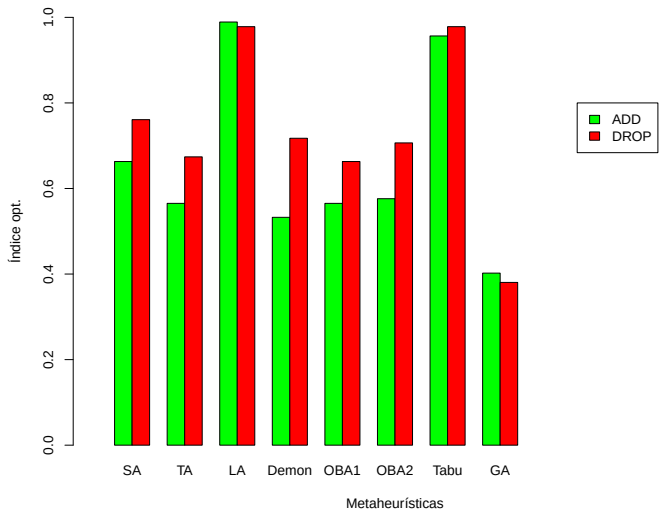

Figura 7.2: Comparativa índice de optimalidad entre las metaheurísticas estudiadas.

Apéndice A

Tabla métodos exactos

Cuadro A.1: Tabla de métodos exactos

Índice	Datos	Tamaño	Método	Z_LP	nº p.c	nº var.	nº rest.	Óptimo	Cota Inf.	Cota Sup.	TE
1	cap101	25x50	weak	659 341.15	-	1275	75	796 648.44			0.015
			strong	796 648.44	-	1275	1300	796 648.44			0.016
			CP	796 648.44	75	1275	150	796 648.44			0.005
			B&C	785 042.16	-	-	-	796 648.44			0.070
2	cap102	25x50	weak	664 015.90	-	1275	75	854 704.20			0.022
			strong	854 704.20	-	1275	1300	854 704.20			0.013
			CP	854 704.20	94	1275	169	854 704.20			0.007
			B&C	853 184.55	-	-	-	854 704.20			0.094
3	cap103	25x50	weak	668 503.01	-	1275	75	893 782.11			0.024
			strong	893 782.11	-	1275	1300	893 782.11			0.014
			CP	893 782.11	116	1275	191	893 782.11			0.009
			B&C	888 784.90	-	-	-	893 782.11			0.130
4	cap104	25x50	weak	674 734.59	-	1275	75	928 941.75			0.027
			strong	928 941.75	-	1275	1300	928 941.75			0.014
			CP	928 941.75	127	1275	202	928 941.75			0.010
			B&C	928 905.36	-	-	-	928 941.75			0.055
5	cap111	50x50	weak	631 421.45	-	2550	100	793 439.56			0.037
			strong	793 439.56	-	2550	2550	793 439.56			0.031
			CP	793 439.56	144	2550	244	793 439.56			0.033
			B&C	793 436.29	-	-	-	793 439.56			0.114
6	cap112	50x50	weak	636 321.45	-	2550	100	851 495.33			0.061
			strong	851 495.33	-	2550	2550	851 495.33			0.030
			CP	851 495.33	192	2550	292	851 495.33			0.041
			B&C	851 282.41	-	-	-	851 495.33			0.153
7	cap113	50x50	weak	641 221.45	-	2550	100	893 076.71			0.075
			strong	893 076.71	-	2550	2550	893 076.71			0.032
			CP	893 076.71	243	2550	343	893 076.71			0.045
			B&C	891 585.16	-	-	-	893 076.71			0.075
8	cap114	50x50	weak	648 426.30	-	2550	100	928 941.75			0.082
			strong	928 941.75	-	2550	2550	928 941.75			0.030
			CP	928 941.75	254	2550	354	928 941.75			0.058
			B&C	928 810.76	-	-	-	928 941.75			0.075
9	cap121	50x50	weak	631 421.45	-	2550	100	793 439.56			0.036
			strong	793 439.56	-	2550	2550	793 439.56			0.031
			CP	793 439.56	144	2550	244	793 439.56			0.032
			B&C	793 436.29	-	-	-	793 439.56			0.105
10	cap122	50x50	weak	636 321.45	-	2550	100	851 495.33			0.072
			strong	851 495.33	-	2550	2550	851 495.33			0.031
			CP	851 495.33	192	2550	292	851 495.33			0.043
			B&C	851 282.41	-	-	-	851 495.33			0.151
11	cap123	50x50	weak	641 221.45	-	2550	100	893 076.71			0.080
			strong	893 076.71	-	2550	2550	893 076.71			0.032
			CP	893 076.71	243	2550	343	893 076.71			0.047
			B&C	891 585.16	-	-	-	893 076.71			0.075
12	cap124	50x50	weak	648 426.30	-	2550	100	928 941.75			0.081
			strong	928 941.75	-	2550	2550	928 941.75			0.030
			CP	928 941.75	254	2550	354	928 941.75			0.058
			B&C	928 810.76	-	-	-	928 941.75			0.077
13	cap131	50x50	weak	631 421.45	-	2550	100	793 439.56			0.037
			strong	793 439.56	-	2550	2550	793 439.56			0.030
			CP	793 439.56	144	2550	244	793 439.56			0.031
			B&C	793 436.29	-	-	-	793 439.56			0.107
14	cap132	50x50	weak	636 321.45	-	2550	100	851 495.33			0.059
			strong	851 495.33	-	2550	2550	851 495.33			0.028
			CP	851 495.33	192	2550	292	851 495.33			0.043
			B&C	851 282.41	-	-	-	851 495.33			0.154
15	cap133	50x50	weak	641 221.45	-	2550	100	893 076.71			0.071

			strong	893 076.71	-	2550	2550	893 076.71			0.030
			CP	893 076.71	243	2550	343	893 076.71			0.045
			B&C	891 585.16	-	-	-	893 076.71			0.076
16	cap134	50x50	weak	648 426.30	-	2550	100	928 941.75			0.082
			strong	928 941.75	-	2550	2550	928 941.75			0.030
			CP	928 941.75	254	2550	354	928 941.75			0.060
			B&C	928 810.76	-	-	-	928 941.75			0.074
17	cap41	16x50	weak	844 807.59	-	816	66	932 615.75			0.007
			strong	932 615.75	-	816	850	932 615.75			0.006
			CP	932 615.75	59	816	125	932 615.75			0.004
			B&C	927 459.80	-	-	-	932 615.75			0.087
18	cap42	16x50	weak	849 169.04	-	816	66	977 799.40			0.009
			strong	977 799.40	-	816	850	977 799.40			0.007
			CP	977 799.40	69	816	135	977 799.40			0.005
			B&C	975 472.56	-	-	-	977 799.40			0.091
19	cap43	16x50	weak	853 434.98	-	816	66	1 010 641.45			0.013
			strong	1 010 641.45	-	816	850	1 010 641.45			0.006
			CP	1 010 641.45	89	816	155	1 010 641.45			0.006
			B&C	1 006 689.52	-	-	-	1 010 641.45			0.098
20	cap44	16x50	weak	859 463.45	-	816	66	1 034 976.98			0.016
			strong	1 034 976.98	-	816	850	1 034 976.98			0.007
			CP	1 034 976.98	98	816	164	1 034 976.98			0.006
			B&C	1 034 792.49	-	-	-	1 034 976.98			0.117
21	cap51	16x50	weak	853 434.98	-	816	66	1 010 641.45			0.014
			strong	1 010 641.45	-	816	850	1 010 641.45			0.007
			CP	1 010 641.45	89	816	155	1 010 641.45			0.006
			B&C	1 006 689.52	-	-	-	1 010 641.45			0.101
22	cap61	16x50	weak	844 807.59	-	816	66	932 615.75			0.007
			strong	932 615.75	-	816	850	932 615.75			0.006
			CP	932 615.75	59	816	125	932 615.75			0.004
			B&C	927 459.80	-	-	-	932 615.75			0.094
23	cap62	16x50	weak	849 169.04	-	816	66	977 799.40			0.010
			strong	977 799.40	-	816	850	977 799.40			0.006
			CP	977 799.40	69	816	135	977 799.40			0.005
			B&C	975 472.56	-	-	-	977 799.40			0.093
24	cap63	16x50	weak	853 434.98	-	816	66	1 010 641.45			0.012
			strong	1 010 641.45	-	816	850	1 010 641.45			0.006
			CP	1 010 641.45	89	816	155	1 010 641.45			0.006
			B&C	1 006 689.52	-	-	-	1 010 641.45			0.101
25	cap64	16x50	weak	859 463.45	-	816	66	1 034 976.98			0.015
			strong	1 034 976.98	-	816	850	1 034 976.98			0.007
			CP	1 034 976.98	98	816	164	1 034 976.98			0.006
			B&C	1 034 792.49	-	-	-	1 034 976.98			0.119
26	cap71	16x50	weak	844 807.59	-	816	66	932 615.75			0.007
			strong	932 615.75	-	816	850	932 615.75			0.007
			CP	932 615.75	59	816	125	932 615.75			0.004
			B&C	927 459.80	-	-	-	932 615.75			0.087
27	cap72	16x50	weak	849 169.04	-	816	66	977 799.40			0.009
			strong	977 799.40	-	816	850	977 799.40			0.006
			CP	977 799.40	69	816	135	977 799.40			0.005
			B&C	975 472.56	-	-	-	977 799.40			0.092
28	cap73	16x50	weak	853 434.98	-	816	66	1 010 641.45			0.012
			strong	1 010 641.45	-	816	850	1 010 641.45			0.006
			CP	1 010 641.45	89	816	155	1 010 641.45			0.006
			B&C	1 006 689.52	-	-	-	1 010 641.45			0.097
29	cap74	16x50	weak	859 463.45	-	816	66	1 034 976.98			0.015
			strong	1 034 976.98	-	816	850	1 034 976.98			0.008
			CP	1 034 976.98	98	816	164	1 034 976.98			0.006
			B&C	1 034 792.49	-	-	-	1 034 976.98			0.118
30	cap81	25x50	weak	659 341.15	-	1275	75	796 648.44			0.014
			strong	796 648.44	-	1275	1300	796 648.44			0.013
			CP	796 648.44	75	1275	150	796 648.44			0.005
			B&C	785 042.16	-	-	-	796 648.44			0.070
31	cap82	25x50	weak	664 015.90	-	1275	75	854 704.20			0.020
			strong	854 704.20	-	1275	1300	854 704.20			0.013
			CP	854 704.20	94	1275	169	854 704.20			0.007
			B&C	853 184.55	-	-	-	854 704.20			0.093
32	cap83	25x50	weak	668 503.01	-	1275	75	893 782.11			0.022
			strong	893 782.11	-	1275	1300	893 782.11			0.014
			CP	893 782.11	116	1275	191	893 782.11			0.008
			B&C	888 784.90	-	-	-	893 782.11			0.131
33	cap84	25x50	weak	674 734.59	-	1275	75	928 941.75			0.028
			strong	928 941.75	-	1275	1300	928 941.75			0.015
			CP	928 941.75	127	1275	202	928 941.75			0.010
			B&C	928 905.36	-	-	-	928 941.75			0.054
34	cap91	25x50	weak	659 341.15	-	1275	75	796 648.44			0.012
			strong	796 648.44	-	1275	1300	796 648.44			0.014
			CP	796 648.44	75	1275	150	796 648.44			0.006
			B&C	785 042.16	-	-	-	796 648.44			0.071
35	cap92	25x50	weak	664 015.90	-	1275	75	854 704.20			0.019
			strong	854 704.20	-	1275	1300	854 704.20			0.013
			CP	854 704.20	94	1275	169	854 704.20			0.007
			B&C	853 184.55	-	-	-	854 704.20			0.094
36	cap93	25x50	weak	668 503.01	-	1275	75	893 782.11			0.026
			strong	893 782.11	-	1275	1300	893 782.11			0.013
			CP	893 782.11	116	1275	191	893 782.11			0.008
			B&C	888 784.90	-	-	-	893 782.11			0.127
37	cap94	25x50	weak	674 734.59	-	1275	75	928 941.75			0.028
			strong	928 941.75	-	1275	1300	928 941.75			0.014
			CP	928 941.75	127	1275	202	928 941.75			0.010
			B&C	928 905.36	-	-	-	928 941.75			0.055
38	capa	100x1000	weak	4 801 483.12	-	100100	1100	22 877 645.84	8 725 280	22 877 645.84	60.193
			strong	17 156 454.48	-	100100	101000	17 156 454.48			2.094
			CP	0	5000	100100	6100	23 964 722.50	10 338 287	23 964 722.50	63.445

			B&C	16 373 422.04	-	-	-	22 821 578.85	16 373 423	22 821 578.85	59.129
39	capb	100x1000	weak	3 668 833.37	-	100100	1100	16 060 512.94	7 412 283	16 060 512.94	59.538
			strong	12 979 071.58	-	100100	101000	12 979 071.58			1.976
			CP	0	5000	100100	6100	13 731 847.82	9 315 938	13 731 847.82	64.865
			B&C	12 979 018.78	-	-	-	12 979 071.58			54.962
40	capc	100x1000	weak	3 300 869.23	-	100100	1100	14 933 075.33	7 127 389	14 933 075.33	60.300
			strong	11 500 104.96	-	100100	101000	11 505 594.33			3.954
			CP	0	5000	100100	6100	12 427 035.04	8 751 960	12 427 035.04	65.061
			B&C	11 505 286.34	-	-	-	11 505 594.33			54.810
41	MO1	100x100	weak	705.23	-	10100	200	1314.74	1133	1314.74	59.472
			strong	1267.06	-	10100	10100	1305.95			10.166
			CP	0	3281	10100	3481	1305.95			7.925
			B&C	1311.65	-	-	-	1305.95			10.018
42	MO2	100x100	weak	790.16	-	10100	200	1446.11	1301	1446.11	59.799
			strong	1383.67	-	10100	10100	1432.36			9.116
			CP	0	3209	10100	3409	1432.36			7.562
			B&C	1422.17	-	-	-	1432.36			10.418
43	MO3	100x100	weak	810.52	-	10100	200	1516.77	1285	1516.77	59.994
			strong	1467.45	-	10100	10100	1516.77			9.250
			CP	0	3482	10100	3682	1516.77			10.437
			B&C	1521.92	-	-	-	1516.77			16.706
44	MO4	100x100	weak	790.34	-	10100	200	1442.24	1286	1442.24	59.576
			strong	1417.26	-	10100	10100	1442.24			10.340
			CP	0	3235	10100	3435	1442.24			7.835
			B&C	1440.17	-	-	-	1442.24			6.100
45	MO5	100x100	weak	755.14	-	10100	200	1411.11	1216	1411.11	59.630
			strong	1367.72	-	10100	10100	1408.77			6.662
			CP	0	3419	10100	3619	1408.77			10.004
			B&C	1416.60	-	-	-	1408.77			8.646
46	MP1	200x200	weak	1470.30	-	40200	400	2864.19	1711	2864.19	59.465
			strong	2586.79	-	40200	40200	2686.48			46.301
			CP	0	5000	40200	5400	2825.14	2116	2825.14	66.421
			B&C	2581.06	-	-	-	3895.73	2582	3895.73	59.609
47	MP2	200x200	weak	1540.15	-	40200	400	3058.04	1767	3058.04	59.594
			strong	2782.00	-	40200	40200	2904.86			49.508
			CP	0	5000	40200	5400	3098.15	2173	3098.15	63.425
			B&C	2761.63	-	-	-	3955.38	2762	3955.38	59.965
48	MP3	200x200	weak	1432.36	-	40200	400	2807.58	1664	2807.58	59.501
			strong	2549.16	-	40200	40200	2623.71			37.465
			CP	0	5000	40200	5400	2798.14	2048	2798.14	62.920
			B&C	2532.32	-	-	-	3749.77	2533	3749.77	59.985
49	MP4	200x200	weak	1555.40	-	40200	400	3083.41	1806	3083.41	59.282
			strong	2805.94	-	40200	40200	2938.75			50.256
			CP	0	5000	40200	5400	3020.29	2224	3020.29	63.232
			B&C	2793.39	-	-	-	4083.22	2794	4083.22	59.971
50	MP5	200x200	weak	1539.29	-	40200	400	3059.12	1777	3059.12	59.692
			strong	2764.77	-	40200	40200	2932.33			57.906
			CP	0	5000	40200	5400	3169.64	2191	3169.64	62.901
			B&C	2759.23	-	-	-	4034.46	2760	4034.46	59.977
51	MQ1	300x300	weak	2176.93	-	90300	600	4960.24	2382	4960.24	60.060
			strong	3943.53	-	90300	90300	4091.01	3947	4091.01	75.732
			CP	2669.58	5001	90300	5400	4980.68	2686	4980.68	64.743
			B&C	3471.06	-	-	-	5797.51	3472	5797.51	59.948
52	MQ2	300x300	weak	2181.41	-	90300	600	4711.92	2397	4711.92	60.204
			strong	3877.54	-	90300	90300	4028.33	3883	4028.33	76.661
			CP	2680.47	5001	90300	5400	5044.71	2701	5044.71	64.984
			B&C	3453.58	-	-	-	5886.78	3454	5886.78	59.953
53	MQ3	300x300	weak	2268.22	-	90300	600	5055.50	2475	5055.50	61.845
			strong	4128.38	-	90300	90300	4275.43	4133	4275.43	75.971
			CP	2763.97	5001	90300	5400	5713.78	2783	5713.78	65.016
			B&C	3553.94	-	-	-	5969.89	3554	5969.89	59.950
54	MQ4	300x300	weak	2252.94	-	90300	600	4776.74	2476	4776.74	60.030
			strong	4065.97	-	90300	90300	4235.15	4070	4235.15	76.841
			CP	2773.40	5001	90300	5400	5319.19	2793	5319.19	65.839
			B&C	3593.70	-	-	-	6005.53	3594	6005.53	59.921
55	MQ5	300x300	weak	2208.51	-	90300	600	4743.70	2417	4743.70	60.143
			strong	3859.74	-	90300	90300	4118.80	3863	4118.80	77.208
			CP	2723.67	5001	90300	5400	4980.78	2744	4980.78	64.932
			B&C	3504.55	-	-	-	6021.24	3505	6021.24	59.945
56	MR1	500x500	weak	1026.50	-	250500	1000	4347.15	1148	4347.15	61.571
			strong	2435.53	-	250500	250500	3567.31	2440	3567.31	119.526
			CP	0	5000	250500	6000	4180.50	1251	4180.50	75.995
			B&C	1412.16	-	-	-	3604.29	1413	3604.29	59.813
57	MR2	500x500	weak	1031.67	-	250500	1000	4268.78	1141	4268.78	64.246
			strong	2514.06	-	250500	250500	3453.62	2519	3453.62	119.177
			CP	0	5000	250500	6000	4050.88	1248	4050.88	76.770
			B&C	1405.71	-	-	-	4409.04	1406	4409.04	59.760
58	MR3	500x500	weak	1053.93	-	250500	1000	4386.54	1158	4386.54	61.449
			strong	2586.86	-	250500	250500	3644.81	2594	3644.81	120.171
			CP	0	5000	250500	6000	4286.97	1259	4286.97	75.758
			B&C	1404.79	-	-	-	4479.93	1405	4479.93	59.794
59	MR4	500x500	weak	1057.43	-	250500	1000	4404.04	1176	4404.04	61.315
			strong	2544.71	-	250500	250500	3683.11	2550	3683.11	120.054
			CP	0	5000	250500	6000	4393.32	1282	4393.32	76.611
			B&C	1432.01	-	-	-	4622.63	1433	4622.63	59.797
60	MR5	500x500	weak	993.65	-	250500	1000	4293.29	1112	4293.29	61.031
			strong	2340.88	-	250500	250500	3483.72	2346	3483.72	119.591
			CP	0	5000	250500	6000	4381.22	1223	4381.22	76.463
			B&C	1370.74	-	-	-	3619.47	1371	3619.47	59.808
61	MS1	1000x1000	weak	2054.11	-	1001000	2000	9179.94	2171	9179.94	64.912
			strong	4589.37	-	1001000	1001000	19 558.79	4931	19 558.79	281.501
			CP	0	5000	1001000	7000	9165.87	2263	9165.87	147.382
			B&C	2173.99	-	-	-	9179.94	2174	9179.94	59.772
62	MT1	2000x2000	weak	3854.23	-	4002000	4000	17 832.07	3855	17 832.07	95.770

			strong	3851.13	-	4002000	4002000	33 346.79	9146	33 346.79	1972.240
			CP	3015.25	1001	4002000	4000	17 834.50	3862	17 834.50	123.523
			B&C	0	-	-	-	355 237.79	3855	355 237.79	62.746
63	1023UnifS	100x100	weak	14 028	-	10100	200	71 840			9.852
			strong	67 683.48	-	10100	10100	71 840			8.936
			CP	67 683.48	813	10100	1013	71 840			8.806
			B&C	69 086.47	-	-	-	71 840			6.735
64	1123UnifS	100x100	weak	13 884	-	10100	200	75 063			11.744
			strong	71 385.63	-	10100	10100	75 063			6.799
			CP	71 385.63	772	10100	972	75 063			9.454
			B&C	73 245.75	-	-	-	75 063			9.913
65	1223UnifS	100x100	weak	12 323	-	10100	200	71 756			11.423
			strong	68 269.22	-	10100	10100	71 756			10.153
			CP	68 269.22	759	10100	959	71 756			5.977
			B&C	69 833.62	-	-	-	71 756			6.157
66	123UnifS	100x100	weak	12 271	-	10100	200	71 342			13.360
			strong	67 441.10	-	10100	10100	71 342			7.773
			CP	67 441.10	808	10100	1008	71 342			9.230
			B&C	69 297.47	-	-	-	71 342			7.999
67	1323UnifS	100x100	weak	12 825	-	10100	200	72 414			13.175
			strong	68 287.98	-	10100	10100	72 414			8.168
			CP	68 287.98	770	10100	970	72 414			9.800
			B&C	70 073.28	-	-	-	72 414			7.342
68	1423UnifS	100x100	weak	12 135	-	10100	200	72 632			24.566
			strong	68 966.72	-	10100	10100	72 632			13.012
			CP	68 966.72	806	10100	1006	72 632			7.710
			B&C	70 086.57	-	-	-	72 632			11.471
69	1523UnifS	100x100	weak	14 850	-	10100	200	77 566	76 480	77 566.00	59.537
			strong	72 824.13	-	10100	10100	77 566			40.798
			CP	72 824.13	779	10100	979	77 566			48.881
			B&C	75 000.32	-	-	-	77 566			38.385
70	1623UnifS	100x100	weak	13 192	-	10100	200	74 344			4.855
			strong	72 448.98	-	10100	10100	74 344			3.977
			CP	72 448.98	741	10100	941	74 344			2.777
			B&C	73 476.70	-	-	-	74 344			2.291
71	1723UnifS	100x100	weak	12 571	-	10100	200	73 847			21.885
			strong	69 658.45	-	10100	10100	73 847			7.362
			CP	69 658.45	783	10100	983	73 847			11.095
			B&C	71 427.17	-	-	-	73 847			7.334
72	1823UnifS	100x100	weak	12 066	-	10100	200	71 903			7.946
			strong	69 649.17	-	10100	10100	71 903			4.367
			CP	69 649.17	773	10100	973	71 903			5.720
			B&C	70 741.17	-	-	-	71 903			2.550
73	1923UnifS	100x100	weak	13 864	-	10100	200	75 520			13.876
			strong	71 301.01	-	10100	10100	75 520			10.445
			CP	71 301.01	758	10100	958	75 520			10.823
			B&C	73 551.76	-	-	-	75 520			8.569
74	2023UnifS	100x100	weak	13 910	-	10100	200	72 458			6.035
			strong	70 382.78	-	10100	10100	72 458			4.988
			CP	70 382.78	801	10100	1001	72 458			3.390
			B&C	71 675.47	-	-	-	72 458			2.856
75	2123UnifS	100x100	weak	11 877	-	10100	200	72 631			12.396
			strong	68 643.66	-	10100	10100	72 631			7.363
			CP	68 643.66	786	10100	986	72 631			7.036
			B&C	69 983.22	-	-	-	72 631			6.706
76	2223UnifS	100x100	weak	11 743	-	10100	200	71 945			6.914
			strong	68 550.67	-	10100	10100	71 945			6.078
			CP	68 550.67	775	10100	975	71 945			5.229
			B&C	70 311.65	-	-	-	71 945			5.865
77	223UnifS	100x100	weak	14 012	-	10100	200	74 148			9.186
			strong	71 050.83	-	10100	10100	74 148			6.014
			CP	71 050.83	761	10100	961	74 148			4.488
			B&C	73 819.10	-	-	-	74 148			2.653
78	2323UnifS	100x100	weak	13 370	-	10100	200	71 335			6.441
			strong	68 443.15	-	10100	10100	71 335			4.231
			CP	68 443.15	748	10100	948	71 335			3.272
			B&C	69 758.00	-	-	-	71 335			2.636
79	2423UnifS	100x100	weak	13 429	-	10100	200	75 342			11.113
			strong	71 733.66	-	10100	10100	75 342			7.613
			CP	71 733.66	758	10100	958	75 342			9.683
			B&C	73 060.92	-	-	-	75 342			6.546
80	2523UnifS	100x100	weak	13 382	-	10100	200	73 807			19.039
			strong	69 585.89	-	10100	10100	73 807			10.497
			CP	69 585.89	795	10100	995	73 807			17.540
			B&C	71 234.41	-	-	-	73 807			11.325
81	2623UnifS	100x100	weak	11 958	-	10100	200	70 929			9.629
			strong	68 776.15	-	10100	10100	70 929			4.181
			CP	68 776.15	794	10100	994	70 929			3.549
			B&C	70 840.35	-	-	-	70 929			2.340
82	2723UnifS	100x100	weak	15 882	-	10100	200	77 314			11.484
			strong	73 805.67	-	10100	10100	77 314			8.257
			CP	73 805.67	742	10100	942	77 314			7.233
			B&C	75 992.10	-	-	-	77 314			10.774
83	2823UnifS	100x100	weak	13 593	-	10100	200	75 025			15.112
			strong	71 056.00	-	10100	10100	75 025			8.811
			CP	71 056.00	788	10100	988	75 025			10.708
			B&C	72 303.50	-	-	-	75 025			7.732
84	2923UnifS	100x100	weak	14 069	-	10100	200	75 279			13.880
			strong	71 256.36	-	10100	10100	75 279			8.581
			CP	71 256.36	790	10100	990	75 279			7.202
			B&C	72 764.68	-	-	-	75 279			9.142
85	3023UnifS	100x100	weak	12 170	-	10100	200	75 654			13.788
			strong	71 639.34	-	10100	10100	75 654			11.771
			CP	71 639.34	776	10100	976	75 654			10.693

			B&C	72 567.53	-	-	-	75 654	12.374
86	323UnifS	100x100	weak	13 286	-	10100	200	73 319	16.524
			strong	69 713.09	-	10100	10100	73 319	8.744
			CP	69 713.09	790	10100	990	73 319	8.861
			B&C	70 978.83	-	-	-	73 319	5.530
87	423UnifS	100x100	weak	12 209	-	10100	200	74 979	14.039
			strong	71 194.11	-	10100	10100	74 979	9.494
			CP	71 194.11	793	10100	993	74 979	8.562
			B&C	73 087.96	-	-	-	74 979	6.796
88	523UnifS	100x100	weak	13 322	-	10100	200	69 251	2.620
			strong	67 839.64	-	10100	10100	69 251	2.938
			CP	67 839.64	728	10100	928	69 251	3.078
			B&C	69 610.97	-	-	-	69 251	1.699
89	623UnifS	100x100	weak	12 674	-	10100	200	72 992	10.322
			strong	70 006.39	-	10100	10100	72 992	5.874
			CP	70 006.39	765	10100	965	72 992	6.143
			B&C	72 110.77	-	-	-	72 992	3.003
90	723UnifS	100x100	weak	13 946	-	10100	200	73 709	8.947
			strong	70 790.34	-	10100	10100	73 709	5.887
			CP	70 790.34	768	10100	968	73 709	3.490
			B&C	72 506.70	-	-	-	73 709	5.362
91	823UnifS	100x100	weak	12 225	-	10100	200	71 860	10.090
			strong	68 602.74	-	10100	10100	71 860	7.382
			CP	68 602.74	770	10100	970	71 860	7.227
			B&C	70 161.98	-	0	0	71 860	4.533
92	923UnifS	100x100	weak	12 565	-	10100	200	74 100	14.561
			strong	70 751.36	-	10100	10100	74 100	7.973
			CP	70 751.36	780	10100	980	74 100	6.988
			B&C	72 196.35	-	0	0	74 100	5.787

Apéndice B

Software desarrollado

Se incluye el software desarrollado para este trabajo junto al documento. El directorio *Exacto* contiene los programas *Mosel* con la implementación de los métodos exactos, mientras que en *Aproximado* se encuentra el software desarrollado para los métodos aproximados, junto a los *scripts* de postprocesado. El programa principal ha sido compilado para Windows y Linux, con nombres *win_uflp-heur.exe* y *uflp-heur.exe*, respectivamente.

Bibliografía

Atamtürk, Alper y Martin WP Savelsbergh (2005). «Integer-programming software systems». En: *Annals of operations research* 140.1, págs. 67-124.

Brito, Dionisio Pérez, José Andrés Moreno Pérez y Carlos Gustavo García González (2004). «Búsqueda por entornos variables: Desarrollo y Aplicaciones en localización». En: *Avances en localización de servicios y sus aplicaciones por Blas Pelegrín Pelegrín*, págs. 349-374.

Burke, Edmund K y Yuri Bykov (2012). «The late acceptance hill-climbing heuristic». En: *University of Stirling, Tech. Rep.*

Cornuéjols, Gérard, George Nemhauser y Laurence Wolsey (1983). *The uncapicitated facility location problem.* Inf. téc. Cornell University Operations Research e Industrial Engineering.

Daskin, Mark S (2011). *Network and discrete location: models, algorithms, and applications.* John Wiley & Sons.

Deroussi, Laurent (2016). *Metaheuristics for Logistics.* John Wiley & Sons.

Floudas, Christodoulos A y Panos M Pardalos (2001). *Encyclopedia of optimization.* Vol. 1. Springer Science & Business Media.

Genevois, Mujde Erol, Derya Celik Turkoglu y Michele Cedolin (2018). «Facility Location Problems-A Case Study for ATM Site Selection». En: *Proceedings of The 12th MAC 2018*, pág. 119.

Hu, Te C, Andrew B Kahng y Chung-Wen Albert Tsao (1995). «Old bachelor acceptance: A new class of non-monotone threshold accepting methods». En: *ORSA Journal on Computing* 7.4, págs. 417-425.

Kochetov, Yuri y Dmitry Ivanenko (2005). «Computationally difficult instances for the uncapacitated facility location problem». En: *Metaheuristics: Progress as real problem solvers.* Springer, págs. 351-367.

Michel, Laurent y Pascal Van Hentenryck (2004). «A simple tabu search for warehouse location». En: *European Journal of Operational Research* 157.3, págs. 576-591.

Pirlot, Marc (1996). «General local search methods». En: *European journal of operational research* 92.3, págs. 493-511.

Talbi, El-Ghazali (2009). *Metaheuristics: from design to implementation.* Vol. 74. John Wiley & Sons.

Teixeira, João C y col. «Solving facility location models with modern optimization software: the weak and strong formulations revisited». En: ().

Tohyama, Hiroaki, Kenichi Ida y Jun Matsueda (2011). «A genetic algorithm for the uncapacitated facility location problem». En: *Electronics and Communications in Japan* 94.5, págs. 47-54.

Wood, Ian A y Tom Downs (1998). «Fast optimization by demon algorithms». En:

Yigit, Vecihi, M Emin Aydin y Orhan Turkbey (2004). «Evolutionary simulated annealing algorithms for uncapacitated facility location problems». En: *Adaptive Computing in Design and Manufacture VI*. Springer, págs. 185-194.