Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Sección de Ingeniería Informática

Sistemas Inteligentes en Logística Portuaria

Algoritmo Heurístico para el Problema de Apilado con Tiempos de Espera

Heuristic Algorithm for the Stacking Problem with Waiting Times

Román Weissbacher Rodríguez

Departamento de Ingeniería Informática y de Sistemas

Escuela Superior de Ingeniería y Tecnología

Trabajo de Fin de Grado

La Laguna, 7 de julio de 2015

Doña **María Belén Melián Batista**, con N.I.F. 44.311.040-E profesora Titular de Universidad adscrita al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna, y

D. **Christopher Expósito Izquierdo**, con N.I.F. 78.851.649-J becario de investigación adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna,

C E R T I F I C A N

Que la presente memoria titulada:

"*Sistemas Inteligentes en Logística Portuaria. Algoritmo Heurístico para el Problema de Apilado con Tiempos de Espera*"

ha sido realizada bajo su dirección por D. **Román Weissbacher Rodríguez**, con N.I.F. 78640634-T.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de julio de 2015

Agradecimientos

La realización de esta asignatura de Trabajo de Fin de Grado no hubiera sido posible sin la ayuda de la sección de Ingeniería Informática de la Escuela Superior de Ingeniería y Tecnología que ha posibilitado la realización de este proyecto y se ha encargado de los trámites necesarios.

Por otra parte, agradecer a **María Belén Melián Batista** y a **Christopher Expósito Izquierdo** por su labor en la Dirección de este Proyecto y por haber hecho un control de seguimiento tan exhaustivo y haberme resuelto las dudas en todo momento.

Mención especial para mi familia y amigos, que me han motivado durante la realización de este proyecto para seguir adelante y lograr nuevas metas.

Resumen

El objetivo de este Trabajo de Fin de Grado ha sido integrar los conocimientos adquiridos durante los estudios en el diseño de algoritmos.

En este Trabajo de Fin de Grado se propone la resolución del Stacking Problem with Waiting Times (SP-WT) para el almacenaje y suministro de una serie de bloques para ser puestos a disposición de clientes mediante un algoritmo heurístico.

Para ello, se proponen varias variantes del algoritmo heurístico para llevar a cabo estas tareas y se estudian los resultados obtenidos para determinar las variantes más eficaces según el tamaño del problema.

Abstract

The aim of this Final Degree Project has been to integrate all the knowledge gained during the degree studies in the design of algorithms.

This Final Degree Project proposes the implementation of the Stacking Problem with Waiting Times (SP-WT) to storage and delivery a number of blocks, to be delivered to certain clients.

To do this, several variants are proposed to carry out these tasks and the results are studied to determine the most effective variants depending on the size of the problem.

Índice general

Capítulo 1

Introducción

1.1. Descripción del contexto

La manera más extendida para almacenar objetos homogéneos, ya sean cajas, contenedores, palets, etc. es apilándolos. En este trabajo se hace referencia a estos objetos con el término *bloque*.

En este contexto, el problema se sitúa en un entorno donde se necesite almacenar bloques para satisfacer la necesidad de entrega de los mismos a ciertos clientes. Este entorno da lugar a la aparición de un problema conocido como Stacking Problem, aplicable en múltiples situaciones y entornos, aunque este trabajo se centra en una terminal de contenedores en un puerto marítimo.

Las terminales de contenedores pueden alcanzar un alto grado de automatización gracias a la estandarización del elemento transportado ,el contenedor, el alto nivel de movimientos que se precisan y la importante repercusión que representa la tecnología para la rentabilidad de la terminal.

En este capítulo se describen los problemas que supone almacenar bloques en una serie de pilas y las tareas necesarias para solventarlos para el correcto almacenaje y entrega a los clientes (barcos, portacontenedores y trenes).

Los bloques se sitúan uno encima del otro en una serie de pilas, las cuales tendrán una altura máxima dependiente del campo de aplicación (almacenes, patios de terminales de contenedores, etc.) y sus características físicas (límite de espacio físico).

El apilado condiciona el acceso a los bloques debido a su estructura, Last In First Out (LIFO). Esto es, sólo serán directamente accesibles los bloques situados en la parte más alta de las pilas. Los demás bloques serán accesibles una vez se encuentren en la parte más alta de las pilas en las que están situados. Para ello, será necesario recolocar los bloques situados por encima de éstos. Dicha recolocación, tanto el almacenaje como la extracción de bloques, se efectúa con una grúa de contenedores.

Los bloques serán extraídos de sus ubicaciones debido a que irán siendo requeridos por clientes, por lo que deberán ser accesibles cuando sean requeridos por ellos. En caso de que un bloque no sea accesible al ser requido por un cliente, éste deberá esperar por el mismo hasta que se recoloquen los bloques superiores y sea accesible.

Al recolocar un bloque, se perjudica el acceso de los bloques de la pila en la que se reubica. Sin embargo, si estos bloques tienen un instante de extracción mayor, no supondría ninguna espera. En otro caso, Se retrasaría la extracción de los bloques inferiores, dado que será necesario recolocar los bloques anteriormente reubicados en la pila.

Por lo tanto, la tarea fundamental que hay que hacer es la de decidir dónde van a ir ubicados los bloques que entran en la terminal y dónde se reubicarán los bloques mal ubicados a la hora de extraer algún bloque para que el cliente deba esperar lo menos posible.

1.2. Objetivos

El Stacking Problem busca minimizar el número de movimientos efectuados por las grúas de contenedores, para almacenar y extraer una serie de bloques de un almacén bidimensional. Esto está orientado al beneficio del almacén, debido a que minimizar el número de movimientos efectuados por las grúas de contenedores supone minimizar los costes derivados del uso de las mismas.

Sin embargo, en este Trabajo Fin de Grado se aborda el Stacking Problem with Waiting Times, que por otra parte, busca minimizar los tiempos de espera de los bloques durante su entrega, lo cual está orientado al beneficio del cliente, debido a que minimizar los tiempos de espera de los bloques durante su entrega supone que el cliente debe esperar lo menos posible, lo cual aumenta su grado de satisfacción.

La resolución de este problema persigue determinar la secuencia de movimientos llevados a cabo por una grúa de contenedores para el almacenamiento y extracción de bloques en un almacén. Esta secuencia podrá servir de guía para los operarios responsables de los movimientos llevados a cabo por las grúas, para la mínima espera del cliente.

1.3. Motivación personal

La elección de este proyecto viene dada por el interés personal del alumno, ya que se trata de un problema de logística genérico que puede ser usado en entornos diferentes y para diferentes fines y además se trata de un problema usado en entornos reales, donde es muy importante que las operaciones realizadas sean efectuadas de la mejor manera posible.

1.4. Organización de la memoria

El resto de esta memoria está organizada de la siguiente manera:

Capítulo 2: Se describe el Stacking Problem with Waiting Times y se ilustra mediante un ejemplo.

Capítulo 3: Se presenta el algoritmo heurístico propuesto para resolver el Stacking Problem with Waiting Times y los diferentes elementos que lo componen.

Capítulo 4: Se presenta un conjunto de resultados computacionales obtenidos mediante el algoritmo propuesto durante este trabajo para la resolución del Stacking Problem with Waiting Times.

Capítulo 5: Finalmente, se presentan las principales conclusiones extraídas del presente Trabajo Final de Grado y se proponen varias líneas de trabajo futuras.

Capítulo 2

Stacking Problem with Waiting Times

A continuación, se describe el Stacking Problem with Waiting Times y se presenta un ejemplo gráfico para su correcta comprensión.

2.1. Descripción del problema

El Stacking Problem with Waiting Times (SP-WT) es un problema de optimización NP-duro cuyo objetivo es determinar la secuencia de movimientos llevados a cabo por una grúa de contenedores para almacenar y extraer una serie de bloques homogéneos en un almacén bidimensional a lo largo de un horizonte de planificación H, de manera que los tiempos de espera durante la extracción sean minimizados.

El almacenamiento de los bloques en SP-WT se produce en un almacén bidimensional, el cuál está compuesto por un número de pilas $S = \{1, 2, \ldots, nS\}$ y un número de filas $T = \{1, 2, \ldots, nT\}$, de manera que su capacidad es $M = nS \times nT$. Los bloques pueden situarse en la base de una pila o bien encima de otros bloques. En este problema no se considera el deterioro de los bloques, todos los bloques se consideran iguales físicamente y serán almacenados de la misma manera.

El instante de almacenado de un bloque $c \in C$ en un almacén es denotado como $d(c)$, mientras que el instante de extracción de un bloque de un almacén es denotado como $r(c)$.

Cada bloque $c \in C$ debe ser almacenado en el almacén bidimensional desde que es entregado $d(c) \geq 0$, hasta que deba ser puesto a disposición del cliente $d(c) < r(c) < H$. La pila donde se almacena el bloque c en el almacén es denotado como $s(c) \in S$ mientras que la altura es denotada como $t(c) \in T$. El conjunto de bloques situados sobre un bloque $c \in C$ en el almacén bidimensional es denotado como:

$$O(c) = \{c' \in C \mid (s(c') = s(c)) \wedge (t(c') > t(c))\}. \tag{2.1}$$

Asimismo, el número de bloques almacenados en una pila $s \in S$ es denotado como $h(s)$, mientras que el instante de extracción más pronto de un bloque situado en la pila s es definido como:

$$min(s) = \text{mín}\{r(c) \mid (c \in C) \wedge (s(c) = s)\}. \tag{2.2}$$

El conjunto de pilas del almacén bidimensional donde puede almacenarse un nuevo bloque está compuesto por las pilas que tengan al menos un hueco libre, definido como:

$$\Phi = \{s \in S \mid h(s) < nT\}. \tag{2.3}$$

2.2. Clasificación

Christopher Expósito-Izquierdo et al. [1] han propuesto una clasificación general de los bloques en relación con su tiempo de extracción y dónde han sido almacenados para un problema relacionado llamado Blocks Recolocation Problem (Forster and Bortfeldt [2]). Un bloque mal ubicado es el que se encuentra encima de otro, cuyo instante de extracción es menor. Sea $\Omega(s)$ el conjunto de bloques mal ubicados en la pila s, definido como:

$$\begin{aligned}\Omega(s) = \{c \in C | (s(c) = s) \wedge \exists c' : (s(c') = s) \wedge \\ (t(c') < t(c)) \wedge (r(c') < r(c))\}, \forall\ s \in S.\end{aligned} \tag{2.4}$$

El conjunto de bloques mal ubicados en el almacén bidimensional es definido como:

$$\Omega = \bigcup_{s \in S} \Omega(s). \tag{2.5}$$

Además, un bloque bien ubicado es el que no está almacenado encima de ningún otro bloque con un tiempo de extracción menor en la misma pila. Son definidos como:

$$\Upsilon(s) = \{c \in C \mid (s(c) = s) \wedge (c \notin \Omega(s))\}, \forall\ s \in S. \tag{2.6}$$

El conjunto de pilas compuesto por únicamente bloques bien ubicados es definido como:

$$\chi = \{s \in S \mid !\exists\ c \in C : (s(c) = s) \wedge (c \notin \Upsilon(s))\}. \tag{2.7}$$

Los movimientos posibles a ser efectuados por la grúa de contenedores son descritos como sigue:

- *Movimiento de inserción.* El siguiente bloque a almacenar es ubicado en la parte más alta de una de las pilas en Φ en su instante de almacenado. Una vez almacenado, el número de bloques del almacén se incrementa en una unidad.

- *Movimiento de extracción.* El siguiente bloque a extraer es retirado de lo más alto de su pila en su instante de extracción. Una vez extraído el bloque, el número de bloques del almacén se decrementa en una unidad. En este trabajo, se prioriza la extracción, de manera que si en un instante de tiempo hay que realizar una inserción y una extracción, se realizará la extracción. La inserción tendrá que efectuarse en el instante de tiempo siguiente, en caso de que no haya que efectuar otra extracción.

- *Movimiento de recolocación.* Uno de los bloques del almacén bidimensional es movido a lo más alto de otra pila con al menos un hueco libre, por lo que no hay cambios en el número de bloques del almacén. Éste movimiento se efectuará sobre un bloque que se encuentre encima de otro que deba ser extraído antes.

Cada solución para el SP-WT está compuesta por nC movimientos de inserción y mC movimientos de extracción con objetivo de almacenar y extraer bloques en el almacén, respectivamente. Los movimientos de recolocación sólo serán necesarios en caso de que un bloque a extraer no sea accesible.

2.3. Ejemplo

A continuación se puede ver un ejemplo ilustrativo del estado de un almacén compuesto por $nS = 8$ pilas y $nT = 5$ filas en el instante de tiempo $t = 16$, donde los bloques con tiempo de extracción antes de $t = 16$ ya han sido extraídos del almacén.

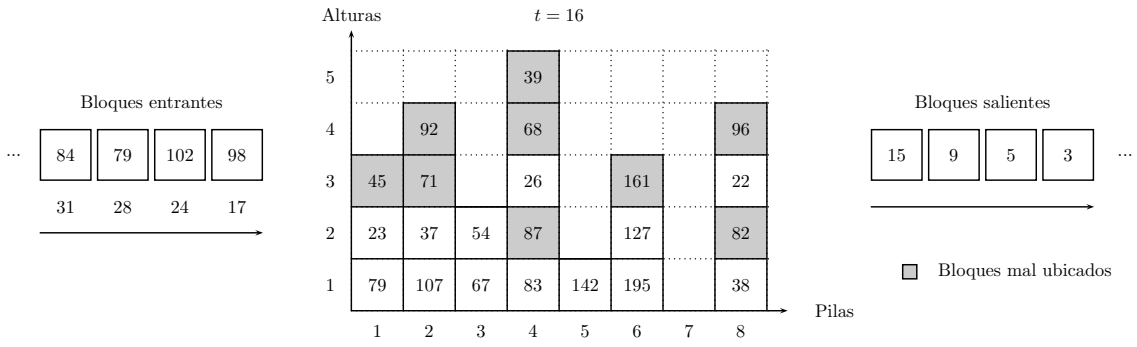

Figura 2.1: Ejemplo del Stacking Problem con $nS = 8$ pilas y $nT = 5$ filas

En este momento, el almacén contiene 22 bloques, donde la pila 4 no tiene huecos libres, por lo que bloques entrantes no pueden ser almacenados en esta pila. Por otra parte, la pila 7 está vacía, por lo que como mucho pueden ser almacenados nT bloques en ella. Como se puede comprobar, cada bloque tiene un instante de extracción que establece un orden relativo de salida de los bloques. Por ejemplo, el próximo bloque a almacenar se almacenará en el instante 17 y deberá ser extraído en el instante 98. Los bloques son identificados por su instante de extracción, por lo que cuando se hace referencia al bloque 98, se hace referencia a un bloque que deberá ser extraido en el instante 98.

Se sabe que en los instantes de tiempo 17, 24, 28 y 31 se deben almacenar los bloques 98, 102, 79 y 84 y que se deben extraer los bloques del almacén en su instante de extracción. Por lo tanto, hay que procurar que los movimientos de inserción y extracción se efectúen en su instante tiempo correspondiente.

En cada instante de tiempo se comprueba si hay que efectuar algún movimiento de inserción o extracción, en cuyo caso se procederá con el movimiento de inserción o extracción correspondiente. En caso de que no haya que efectuar ninguno de estos dos movimientos, se comprueba si los próximos bloques a extraer son accesibles; y si no es así, se recoloca un bloque que impida la extracción del próximo bloque a extraer que no se encuentre en lo más alto de una pila.

Las soluciones factibles del problema son secuencias de t-uplas de la forma (a, b, c, d) que se describen como:

1. a, instante de tiempo de retirada de bloque
2. b, pila de origen
3. c, pila de destino
4. d, instante de tiempo de ejecución de movimiento

En caso de haber un 0 como pila de origen, se indica que es una inserción y en caso de haber un 0 como pila de destino, se indica que es una extracción.

- Inserción: (10, 0, 1, 1). Se inserta el bloque 10 en la pila 1, en el instante 1.
- Recolocación: (10, 1, 2, 2). Se recoloca el bloque 10 de la pila 1 a la pila 2, en el instante 2.
- Extracción: (10, 2, 0, 3). Se extrae el bloque 10 de la pila 2, en el instante 3.

En este Trabajo Fin de Grado se usa un algoritmo heurístico que compara el uso de diferentes estrategias que hacen que se alcancen diferentes resultados, las cuales serán descritas de manera exhaustiva en el Capítulo 3. A continuación se muestra una solución factible parcial hasta el instante de tiempo t_{30} usando una estrategia que efectúa movimientos aleatorios con motivo de entender mejor el comportamiento del algoritmo usado:

- t_{16}: (96, 8, 5, 16). Se recoloca el bloque 96 de la pila 8 a la 5, para hacer accesible el bloque 22, que es el siguiente bloque a extraer.

- t_{17}: (98, 0, 7, 17). Se inserta el bloque 98 en la pila 7.

- t_{18}: (45, 1, 5, 18). Se recoloca el bloque 45 de la pila 1 a la 5, para hacer accesible el bloque 23, ya que el bloque 22 ya se encuentra accesible.

- t_{19}: (39, 4, 2, 19). Se recoloca el bloque 39 de la pila 4 a la 2, para hacer accesible el bloque 26, ya que los bloques 22 y 23 ya se encuentran accesibles.

- t_{20}: (68, 4, 3, 20). Se recoloca el bloque 68 de la pila 4 a la 3, para hacer accesible el bloque 26, ya que los bloques 22 y 23 ya se encuentran accesibles.

- t_{21}: (39, 2, 5, 21). Se recoloca el bloque 39 de la pila 2 a la 5, para hacer accesible el bloque 37, ya que los bloques 22, 23 y 26 ya se encuentran accesibles.

- t_{22}: (22, 8, 0, 22). Se extrae el bloque 22 de la pila 8.

- t_{23}: (23, 1, 0, 23). Se extrae el bloque 23 de la pila 1.

- t_{24}: (102, 0, 3, 24). Se inserta el bloque 102 en la pila 3.

- t_{25}: (92, 2, 7, 25). Se recoloca el bloque 92 de la pila 2 a la 7, para hacer accesible el bloque 37, ya que el bloque 26 ya se encuentra accesible.

- t_{26}: (26, 4, 0, 26). se extrae el bloque 26 de la pila 4.

- t_{27}: (71, 2, 6, 27). Se recoloca el bloque 71 de la pila 2 a la 6, para hacer accesible el bloque 37.

- t_{28}: (79, 0, 1, 28). Se inserta el bloque 79 en la pila 1.

- t_{29}: (82, 8, 4, 29). Se recoloca el bloque 82 de la pila 8 a la 4, para hacer accesible el bloque 38, ya que el bloque 37 ya se encuentra accesible.

- t_{30}: (102, 3, 1, 30). Se recoloca el bloque 102 de la pila 3 a la 1, para hacer accesible el bloque 54, ya que los bloques 37, 38, 39 y 45 ya se encuentran accesibles.

Capítulo 3

Algoritmo Heurístico

En este capítulo se describe el algoritmo heurístico usado para la resolución del Stacking Problem with Waiting Times.

El algoritmo usa tres movimientos, correspondientes a los movimientos posibles de la grúa de contenedores, descritos en el capítulo anterior: inserción, extracción y recolocación; y estos movimientos deben efectuarse evitando cualquier espera a la hora de almacenar o extraer algún bloque.

Para que no hayan esperas en las extracciones, los bloques a extraer deben ser accesibles en el instante de su extracción. Esto es, deben estar ubicados en lo más alto de alguna pila, ya que si no es así, habría que reubicar los bloques superiores a éste y existiría un tiempo de espera.

Para que no haya esperas en las inserciones, se deben efectuar de forma correcta las extracciones, ya que si hubo que recolocar un bloque en el instante de tiempo de una extracción y en el instante siguiente se debe efectuar una inserción no se podría hacer, ya que todavía se encuentra pendiente la extracción.

Por lo tanto, para minimizar los tiempos de espera, se debe:

1. Reubicar bloques: En los instantes de tiempo que no se inserten ni se extraigan bloques, se deben reubicar bloques que impidan que los próximos bloques a extraer sean accesibles.

2. Elegir bien las pilas de destino: En los movimientos de inserción y recolocación se debe elegir una pila de destino que lleve a efectuar el menor número posible de movimientos de recolocación en movimientos posteriores.

Como ya hemos mencionado anteriormente, hemos de elegir bien en qué pila deberán ser colocados los bloques a la hora de almacenar o al recolocar un bloque de una pila a otra, por lo que haremos uso de un procedimiento encargado de elegir la pila de destino.

El procedimiento encargado de elegir la pila de destino se divide en dos etapas o fases, una fase de evaluación y otra de selección. La fase de evaluación se encargará de evaluar las pilas, asignando un valor decimal, entre 0 y 1, a cada una de ellas según lo conveniente que sea como pila de destino y la fase de selección se encargará de seleccionar una pila a partir de la evaluación obtenida de la fase de evaluación.

Cada fase tiene cinco métodos o estrategias disponibles, cada una con su propio criterio, que elegirán una pila frente a otra según la adecuación de la pila frente al criterio de la estrategia. Se deberá elegir una sola estrategia de evaluación y una de selección para la obtención de una solución factible. Dichas estrategias se describirán con más detenimiento en las secciones correspondientes de este mismo capítulo.

3.1. Pseudocódigo

A continuación se ilustra el pseudocódigo del algoritmo propuesto para la resolución del Stacking Problem with Waiting Times:

```
1: Leer fichero de instancia
2: Mientras Almacén no lleno y no se ha alcanzado el instante H y
   no se han almacenado y extraído todos los bloques hacer
3:     Si Existe un bloque a extraer  entonces
4:         Si Bloque a extraer es accesible entonces
5:             Extraer bloque
6:         En otro caso
7:             Recolocar bloque superior
8:         fin Si
9:     En otro caso Si Existe un bloque a almacenar entonces
10:        Almacenar bloque en el almacén
11:    En otro caso
12:        Mientras Siguiente bloque a extraer es accesible y Existe
   una pila de destino posible hacer
13:            Mover a siguiente bloque
14:        fin Mientras
15:        Si Siguiente bloque a extraer no es accesible entonces
16:            Recolocar bloque superior
17:        fin Si
18:    fin Si
19: fin Mientras
```

El código de este Trabajo Fin de Grado está programado completamente en Java y su estructura se divide en tres secciones:

- Una sección que contiene las clases base del Proyecto, donde se almacenan todos los datos referentes al problema, pilas, bloques y donde también se encuentra la clase principal del programa.

  Esta sección contiene 9 clases:

  - La clase Main se encarga de la ejecución del programa, lee el fichero de instancia fuente, decide los movimientos a efectuar en cada instante de tiempo y calcula los datos generados.
  - La clase Problem almacena el estado inicial de la ejecución.
  - La clase Solution almacena el estado actual del almacén, tanto como el vector de movimientos efectuados hasta el momento y los bloques restantes por insertar en el almacén.
  - Las clases Block y ComparatorBlock se encargan de almacenar y ordenar los bloques que no hayan sido insertados en el almacén.
  - La clase Move se encarga de almacenar todos los movimientos de inserción, recolocación y extracción efectuados.
  - La clase Pair se encarga de contener los valores obtenidos de las evaluaciones de las pilas, explicado en la siguiente sección.
  - La clase Point sirve de clase auxiliar, la cual consulta los bloques buscados en el almacén y devuelve los que cumplan con el criterio de búsqueda.
  - La clase Result almacena los valores recopilados de las ejecuciones de las diferentes instancias para simplificar la obtención de los datos.

- Una sección que contiene las clases referentes a la fase de evaluación de las pilas, donde se encuentran las estrategias de evaluación y clases necesarias para su funcionamiento. Se hace uso de una interfaz, la cual es implementada por cada una de las clases correspondientes a las diferentes estrategias de evaluación. Esto se hizo así para poder generalizar el código de manera que lo único que cambia a la hora de elegir una estrategia u otra, es la clase creada.

- Una sección que contiene las clases referentes a la fase de selección de las pilas, donde se encuentran las estrategias de selección y las clases necesarias para su funcionamiento. Al igual que con las estrategias de evaluación, se hace uso de una interfaz que es implementada por cada una de las estrategias de selección.

3.2. Fase de evaluación

La fase de evaluación se encarga de evaluar las pilas, asignando un valor decimal, entre 0 y 1, a cada una de ellas según lo conveniente que sea como pila de destino. Cuanto más conveniente sea una pila para ser seleccionada como pila de destino, tendrá un valor más cercano a 1 y cuanto menos conveniente sea la pila, ésta obtendrá un valor más cercano a 0.

Como ya se ha comentado anteriormente, la valoración asignada dependerá de la estrategia de evaluación elegida; y cuanto más conveniente sea una pila de acuerdo al criterio de la estrategia de evaluación elegida, se evaluará con un valor más cercano a 1. Una pila llena, es decir, sin ningún hueco libre para colocar bloques, no puede ser elegida como pila de destino; se le asigna el valor de -1.

Se han implementado cinco estrategias de evaluación, que serán descritas a continuación con más detalle:

- **E1, Estrategia Lowest**: Establecerá un mayor valor a las pilas con menos bloques, de manera que una pila sin bloques tendrá el máximo valor posible, 1. Con esta estrategia se pretende fijar como destino las pilas con menos bloques, con el objetivo de tener un número similar de bloques en todas las pilas y de esta manera pretender obstaculizar lo menos posible la extracción de los bloques. Se obtiene como sigue:

$$f_1(s,T) = \begin{cases} 1 - \frac{h(s)}{nT} \text{ si } 0 \leq h(s) < nT \\ -1 \text{ en otro caso} \end{cases}$$

- **E2, Estrategia Highest**: Establecerá un mayor valor a las pilas con más bloques, de manera que una pila con sólo un espacio libre, tendrá un valor cercano a 1, mientras que una pila vacía tendrá un valor de 0. Esta estrategia pretende fijar como destino las pilas con mayor número de bloques, con el objetivo de tener algunas pilas muy llenas y otras muy vacías, consiguiendo que siempre hayan pilas vacías o con pocos contenedores para ser destino de bloques recolocados. Sin embargo, supone efectuar muchas recolocaciones de bloques. Se obtiene como sigue:

$$f_2(s,T) = \begin{cases} \frac{h(s)}{nT} \text{ si } 0 \leq h(s) < nT \\ -1 \text{ en otro caso} \end{cases}$$

- **E3, Estrategia LowerLater**: Establece un mayor valor a las pilas con menos bloques y cuyos bloques deban ser extraidos lo más tarde posible. De esta forma, si dos pilas tienen el mismo número de bloques, se eligirá la pila cuyos bloques salgan más tarde. Esta estrategia persigue por un lado, obstaculizar al menor número posible de bloques mediante la elección de la pila con menor número de bloques; y por otro lado, elegir la pila que tenga los bloques a extraer más tarde, para que haya mayores probabilidades de que el nuevo bloque a situar se extraiga antes que los ya situados en la pila. Se obtiene como sigue:

$$f_3(s,T) = \begin{cases} \frac{h(s)}{nT} + \frac{min(s)}{1000} \text{ si } 0 < h(s) < nT \\ 1 \text{ si } h(s) = 0 \\ -1 \text{ en otro caso} \end{cases}$$

- **E4, Estrategia LowerEarlier**: Establece un mayor valor a la pila cuyo bloque más próximo a extraer sea ligeramente mayor al bloque a almacenar en la pila, de manera que después de almacenar el bloque, sea éste mismo el más próximo a extraer de esta pila. Esta estrategia persigue elegir la pila que, al introducir el nuevo bloque en la misma, éste quede bien ubicado y que la pila elegida contenga bloques que se extraigan lo más pronto posible. De esta manera, se reservan las demás pilas para bloques cuyo instante de extracción sea mayor. Se obtiene como sigue:

$$f_4(s,c) = \begin{cases} 1 - \frac{min(s)}{1000} \text{ si } min(s) > c \\ 0,5 \text{ si } h(s) = 0 \\ 0,1 \text{ si } min(s) \leq c \\ -1 \text{ en otro caso} \end{cases}$$

- **E5, Estrategia RandomEval**: Calcula un valor aleatorio de los posibles bloques que puede haber en cada pila y le da un mayor valor a las pilas con más bloques, como en la estrategia Lowest. Esta estrategia persigue elegir aleatoriamente la pila de destino con objetivo de comparar con las otras estrategias en caso de existir algún caso en el que sea conveniente una elección aleatoria. Se obtiene como sigue:

$$f_5(s,T) = \begin{cases} 1 - \frac{r(s)}{nT} \text{ si } 0 \leq r(s) < nT \\ -1 \text{ en otro caso} \end{cases}$$

  Siendo r(s) el número aleatorio de bloques en la pila s.

Como resultado de la fase de evaluación se obtienen los valores correspondientes para cada pila, los cuales se ordenan de mayor a menor, formando un vector con la forma $V = (v_1, v_2, ..., v_{nS})$

3.3. Fase de selección

La fase de selección se encarga de seleccionar una pila cogiendo como entrada el vector generado de la fase de evaluación.

Como en la fase de evaluación, tenemos cinco estrategias de selección disponibles. Estas estrategias reciben el vector resultante de la fase de evaluación y seleccionan una pila en base al criterio de la estrategia de selección seleccionada. El vector recibido es ordenado según el criterio de la estrategia de selección, de manera que la pila más adecuada para cada estrategia corresponderá con el valor de la primera posición del vector una vez ordenado.

A continuación se detallan las estrategias de selección:

- **S1, Estrategia Best**: Eligirá la pila con mejor valor, es decir, la pila con el valor más cercano a 1.
- **S2, Estrategia Worst**: Eligirá la pila con peor valor, es decir, la pila con el valor más cercano a 0.
- **S3, Estrategia Weighted**: Calcula un valor aleatorio ponderado de los valores obtenidos de la etapa de evaluación, de manera que una pila con mayor valor que otra, tendrá una mayor probabilidad de ser elegida, pero no deja de ser una selección aleatoria.
- **S4, Estrategia BestRandom**: Elige la pila de destino de forma aleatoria de entre las pilas que tengan el mismo valor que la mejor pila.
- **S5, Estrategia Randomselect**: Establece la pila de destino de forma completamente aleatoria, sin tener en cuenta los valores obtenidos del proceso de evaluación.

Una vez terminada la fase de selección, se conoce la pila de destino y se podrá proceder con el movimiento de inserción o recolocación con dicha pila.

Capítulo 4

Resultados computacionales

En este capítulo se muestran los resultados obtenidos de las diferentes estrategias y una comparativa donde se interpretan los resultados y se explica la lógica de los mismos.

4.1. Tablas de resultados

Para la obtención de los resultados se ha hecho uso de un generador de instancias, generando 800 instancias diferentes entre combinaciones de diferentes tamaños de almacenes y cantidades de bloques.

Los tamaños o configuraciones posibles del problema son de 5, 10, 15 o 20 pilas; 4, 6, 8 o 10 alturas; y 100, 200, 300, 400 o 500 bloques. Cada combinación de pilas, alturas y bloques compone una configuración para la creación de ficheros de instancias, lo cual da lugar a 80 configuraciónes posibles de ficheros.

Los instantes de almacenado y extracción de los bloques en los ficheros de instancias son generados aleatoriamente, por lo que para cada tamaño del problema se generan 10 ficheros, donde cada uno contiene diferentes conjuntos de bloques. Estos 10 ficheros creados por cada uno de los 80 tamaños posibles del problema dan lugar a las 800 instancias comentadas anteriormente.

Las 800 instancias se ejecutan con cada una de las combinaciones de estrategia posibles, dando lugar a una tabla de resultados de 800 filas y 25 columnas. Para la generación de resultados se han tenido que agrupar los resultados calculando la media de los 10 resultados obtenidos de las 10 instancias de cada uno de los tamaños posibles del problema y se han seleccionado los tamaños representativos para instancias pequeñas (5 pilas y 4 alturas), medianas (10 o 15 pilas y 6 u 8 alturas) y grandes (20 pilas y 10 alturas).

Como resultado se ha obtenido una tabla de 30 filas y 25 columnas, donde las filas representan las diferentes instancias resultantes y las columnas las diferentes combinaciones de estrategias usadas.

Por motivos de espacio, la tabla se encuentra dividida en cinco tablas diferentes ($E_1, E_2, ..., E_5$), donde cada tabla representa los resultados de una estrategia de evaluación, con todas las instancias y combinaciones posibles con las estrategias de selección.

Los datos de las instancias son especificados en las tres primeras columnas de las tablas, definiendo nS como número de pilas, nT como número de filas y nC como número de bloques a almacenar y extraer del almacén.

En cuanto al contenido de las tablas, se muestran los instantes de tiempo de espera medios que se necesitaron en las inserciones y extracciones de los bloques durante la ejecución. Por lo tanto, cuanto menor sea el número indicado, correspondiente a cada estrategia, mejor ha sido el resultado de la estrategia para la instancia en cuestión, ya que se han necesitado menos instantes de tiempo de espera durante la ejecución.

Las tablas se encuentran organizadas de esta manera con el objetivo de comparar las estrategias, cuya comparativa se desarrolla en la siguiente seccion.

La última fila represanta el valor medio de cada una de las columnas, con el objetivo de comparar posteriormente las estrategias de una forma más genérica.

E_1, Tabla de estrategia de evaluación Lowest

nS	nT	nC	Best	Worst	Weighted	BestRandom	RandomSelect
5	4	100	43,66	99,73	50,89	43,53	53,94
5	4	200	136,58	188,20	147,75	136,76	146,89
5	4	300	237,50	298,04	244,79	236,89	246,03
5	4	400	336,82	402,94	345,93	336,29	349,49
5	4	500	438,36	499,13	445,98	437,99	448,84
10	6	100	37,11	60,33	43,42	36,22	44,58
10	6	200	128,84	156,31	135,96	128,32	138,58
10	6	300	230,96	258,99	236,30	230,51	238,97
10	6	400	328,30	355,42	337,57	328,85	337,84
10	6	500	429,03	456,46	434,26	428,17	437,51
10	8	100	37,29	77,18	44,14	36,55	47,26
10	8	200	131,69	174,08	138,28	131,63	141,42
10	8	300	230,29	270,65	236,31	229,49	239,49
10	8	400	328,75	373,28	337,57	327,84	338,64
10	8	500	429,20	475,97	436,79	429,66	440,56
15	6	100	32,73	62,28	39,64	33,13	43,23
15	6	200	126,69	159,92	135,12	127,07	136,61
15	6	300	224,77	259,37	232,10	224,49	233,69
15	6	400	324,79	359,62	332,20	324,32	332,81
15	6	500	425,38	458,61	434,56	425,19	434,65
15	8	100	32,64	77,07	41,95	33,20	42,57
15	8	200	127,05	174,01	136,20	126,95	137,60
15	8	300	224,80	275,62	233,06	225,35	233,94
15	8	400	323,26	372,74	332,31	323,73	332,25
15	8	500	423,61	467,45	432,08	423,15	434,12
20	10	100	45,83	76,96	49,85	44,32	49,70
20	10	200	159,28	185,63	162,60	157,68	167,65
20	10	300	258,75	294,35	263,49	259,82	264,53
20	10	400	361,06	393,06	363,64	358,56	368,19
20	10	500	453,53	480,97	457,68	454,91	460,10
			234,95	274,81	242,08	234,69	244,06

E_2, Tabla de estrategia de evaluación Highest

S	T	C	Best	Worst	Weighted	BestRandom	RandomSelect
5	4	100	101,39	43,52	103,77	107,72	53,88
5	4	200	182,39	135,84	187,43	184,10	146,77
5	4	300	302,52	237,66	304,13	303,22	247,73
5	4	400	406,29	336,28	403,09	404,90	348,75
5	4	500	494,10	438,07	505,46	498,19	448,47
10	6	100	60,11	36,79	60,88	59,82	45,83
10	6	200	158,59	128,66	159,69	157,89	138,08
10	6	300	257,91	231,20	258,51	259,41	240,45
10	6	400	357,91	328,37	357,24	359,93	337,40
10	6	500	456,98	428,94	458,85	456,35	438,71
10	8	100	80,10	37,37	76,59	75,36	46,51
10	8	200	171,19	131,61	172,76	173,19	142,92
10	8	300	275,11	230,61	271,55	272,81	239,26
10	8	400	371,56	328,65	375,37	371,98	338,63
10	8	500	471,00	429,48	474,75	476,44	439,31
15	6	100	63,62	32,50	62,12	62,52	41,60
15	6	200	162,14	126,68	160,23	158,73	137,31
15	6	300	260,21	224,93	262,35	260,09	235,71
15	6	400	359,20	324,42	357,92	359,15	333,13
15	6	500	459,12	425,39	460,24	457,92	435,62
15	8	100	74,89	32,85	79,96	76,64	42,06
15	8	200	174,55	127,45	174,96	172,19	137,40
15	8	300	277,36	225,04	274,99	277,63	233,91
15	8	400	371,04	322,98	369,38	370,63	333,03
15	8	500	471,64	423,71	468,92	469,81	434,41
20	10	100	78,72	45,77	76,01	75,64	50,28
20	10	200	188,72	157,68	186,94	182,46	165,89
20	10	300	293,82	259,80	292,00	292,30	265,59
20	10	400	395,79	358,13	394,65	396,23	367,22
20	10	500	479,74	453,75	482,64	479,42	463,04
			275,26	234,80	275,78	275,09	244,30

E_3, Tabla de estrategia de evaluación LowerLater

S	T	C	Best	Worst	Weighted	BestRandom	RandomSelect
5	4	100	41,63	98,58	52,84	41,85	53,14
5	4	200	134,23	189,89	144,32	133,86	145,70
5	4	300	235,94	298,34	243,77	235,84	245,59
5	4	400	334,50	402,27	343,89	334,86	346,21
5	4	500	435,35	500,77	444,43	435,16	448,60
10	6	100	34,02	61,42	43,41	34,29	45,65
10	6	200	125,60	159,47	136,86	125,61	138,19
10	6	300	227,33	262,80	237,72	226,98	239,85
10	6	400	326,32	354,95	335,51	326,49	337,47
10	6	500	425,27	457,85	434,56	425,18	437,54
10	8	100	34,73	79,34	45,25	34,58	48,36
10	8	200	128,15	175,30	139,42	128,47	142,56
10	8	300	227,05	272,23	236,03	226,88	238,62
10	8	400	326,25	375,80	336,10	326,25	340,04
10	8	500	426,86	477,49	438,14	426,58	439,52
15	6	100	29,77	64,08	39,10	29,76	40,73
15	6	200	123,34	161,38	132,61	123,35	136,55
15	6	300	221,81	259,76	231,00	221,64	235,05
15	6	400	321,42	362,44	332,68	321,42	333,83
15	6	500	423,15	461,83	434,36	423,15	435,41
15	8	100	30,71	75,77	40,20	30,73	43,20
15	8	200	125,03	177,13	137,00	124,87	138,06
15	8	300	221,93	276,42	232,93	221,93	234,50
15	8	400	320,90	374,39	331,63	320,72	332,30
15	8	500	421,32	468,14	432,29	421,23	434,14
20	10	100	44,45	73,98	47,62	44,50	49,95
20	10	200	157,54	188,87	163,06	159,81	165,45
20	10	300	255,57	296,37	264,31	255,63	266,33
20	10	400	358,65	394,10	364,69	359,52	370,09
20	10	500	452,73	482,23	457,71	452,26	460,50
			232,39	276,11	241,78	232,45	244,10

E_4, Tabla de estrategia de evaluación LowerEarlier

S	T	C	Best	Worst	Weighted	BestRandom	RandomSelect
5	4	100	39,61	75,33	43,46	37,35	53,82
5	4	200	134,94	169,71	135,79	132,57	146,06
5	4	300	241,91	269,80	235,10	237,13	248,78
5	4	400	336,17	368,56	333,33	336,22	346,31
5	4	500	438,08	472,52	436,21	436,33	450,07
10	6	100	33,01	66,43	35,26	31,75	46,37
10	6	200	133,72	160,47	127,44	131,14	137,33
10	6	300	231,70	261,30	228,53	231,02	239,14
10	6	400	331,59	361,55	327,84	329,91	337,00
10	6	500	431,97	461,86	427,34	430,13	439,08
10	8	100	35,34	74,15	35,82	35,54	47,75
10	8	200	136,67	166,33	130,57	132,56	142,29
10	8	300	235,87	266,48	228,45	234,41	238,67
10	8	400	338,26	369,23	328,56	334,29	338,59
10	8	500	433,73	468,33	427,92	433,02	440,06
15	6	100	32,41	67,99	31,75	31,37	42,05
15	6	200	131,55	164,60	127,69	132,28	135,63
15	6	300	233,32	261,91	224,64	231,47	233,31
15	6	400	332,38	362,77	324,18	332,04	333,88
15	6	500	434,83	462,39	426,11	433,51	435,22
15	8	100	34,08	70,13	30,92	33,66	42,53
15	8	200	136,35	167,49	127,51	135,50	139,48
15	8	300	233,78	264,12	224,59	233,24	234,78
15	8	400	335,03	364,53	323,94	334,51	333,96
15	8	500	430,92	463,82	424,40	429,55	434,23
20	10	100	49,80	65,83	46,05	42,30	50,92
20	10	200	162,82	176,73	158,00	161,49	166,60
20	10	300	265,23	273,71	258,63	233,32	267,36
20	10	400	364,88	374,59	362,48	360,96	371,11
20	10	500	452,09	470,40	450,60	452,03	461,84
			238,73	267,44	234,10	236,02	244,47

E_5, Tabla de estrategia de evaluación RandomEval

S	T	C	Best	Worst	Weighted	BestRandom	RandomSelect
5	4	100	54,18	55,23	52,51	52,84	57,58
5	4	200	148,78	146,53	145,66	145,58	145,67
5	4	300	247,42	247,73	248,53	247,75	245,79
5	4	400	348,28	348,54	345,50	345,59	348,59
5	4	500	448,42	449,32	447,23	450,41	447,63
10	6	100	50,04	46,77	47,04	43,84	45,78
10	6	200	141,21	142,87	139,11	138,30	138,23
10	6	300	244,08	245,33	239,69	240,26	241,43
10	6	400	341,37	343,55	338,46	336,39	337,65
10	6	500	441,23	442,79	438,16	436,05	438,05
10	8	100	50,49	51,22	48,60	46,66	48,65
10	8	200	146,33	144,82	143,53	141,50	140,20
10	8	300	243,17	242,90	237,98	241,22	239,59
10	8	400	344,61	342,47	338,95	339,16	339,77
10	8	500	443,82	443,09	440,09	438,49	439,54
15	6	100	49,61	49,55	44,20	41,74	42,15
15	6	200	146,95	144,68	137,56	135,63	137,48
15	6	300	243,67	241,09	235,72	233,63	234,45
15	6	400	341,53	342,35	334,47	334,01	334,78
15	6	500	440,10	443,89	433,16	433,79	434,88
15	8	100	48,01	48,22	41,64	42,99	42,57
15	8	200	144,55	144,19	138,16	137,96	139,77
15	8	300	240,58	240,56	234,96	234,37	234,78
15	8	400	339,68	338,00	332,49	331,47	331,85
15	8	500	440,70	440,04	433,05	435,64	433,54
20	10	100	50,11	58,09	50,14	50,82	60,75
20	10	200	160,44	164,21	166,02	163,28	165,30
20	10	300	269,12	267,69	265,23	265,19	263,45
20	10	400	370,23	368,68	366,32	364,80	372,29
20	10	500	462,59	460,20	461,73	458,91	460,44
			248,04	248,15	244,20	243,61	244,75

La ejecución de todas las instancias ha tardado 5 minutos y 44 segundos, usando una máquina con las siguientes características, cuyos datos fueron tomados antes de la ejecución:

Operating System:	Windows 7 64-bit (Build 7601) Service Pack 1
System Manufacturer:	ASUSTeK Computer Inc.
System Model:	K52JU
Processor:	Intel(R) Core(TM) i3 M 370 2.40GHz (4 CPUs)
Memory:	4096MB RAM
Available OS Memory:	3948MB RAM
Page File:	3395MB used, 4500MB available

4.2. Comparativa

Cada estrategia tiene una forma distinta de evaluar o seleccionar y la combinación de estrategias da lugar a la existencia de muchos valores diferentes. Al comportarse cada estrategia de una manera diferente, se obtendrán mejores resultados para cierto tipo de instancias. En este apartado se analizan los resultados obtenidos en las tablas de la sección anterior para identificar las estrategias que funcionan mejor con cada una de las instancias.

En cuanto a las tablas E_1, E_3 y E_4, correspondientes a las estrategias de evaluación Lowest, LowerLater y LowerEarlier, se puede observar que los mejores resultados se alcanzan con las estrategias de selección Best y BestRandom. Por otro lado, las tablas E_2 y E_5, correspondientes a las estrategias de evaluación Highest y RandomEval, no alcanzan los mejores resultados con estas mismas estrategias de selección. En el caso de Highest, el mejor resultado se alcanza con la estrategia de selección Worst; y en el caso de RandomEval, al tratarse de una estrategia completamente aleatoria, todas las combinaciones con estrategias de selección aportan resultados similares, por lo que a partir de ahora la comparativa se enfoca en las demás estrategias de evaluación.

Tratando de identificar las estrategias de evaluación que mejor funcionan, se comparan los resultados obtenidos entre la mejor y la peor valoración de las pilas, ya que, si se ha hecho una buena valoración de las pilas, estos dos valores deberían tener valores bastante distanciados entre sí. A continuación se muestran los valores promedios de las columnas Best y Worst de las cinco tablas:

E / S	Best	Worst
Lowest	234,95	274,81
Highest	275,26	234,80
LowerLater	232,39	276,11
LowerEarlier	238,73	267,44
RandomEval	248,04	248,15

Como se puede ver en esta última tabla, los valores de las columnas Best y Worst para la estrategia de evaluación RandomEval tiene valores semejantes, por lo que no hace una buena evaluación de las pilas, por lo comentado anteriormente, ya que se trata de una estrategia aleatoria. Por otra parte, la estrategia de evaluación Highest sólo obtiene buenos resultados con la estrategia de selección Worst, lo cual es lógico ya que la estrategia Highest crea la necesidad de efectuar muchas recolocaciones al tener pilas con muchos bloques, por lo que la estrategia de selección Worst elige la peor pila según el criterio Highest, lo cual es equivalente a la mejor pila de la estrategia Lowest. Por lo tanto, a partir de ahora la comparativa irá enfocada a las tres tablas restantes, las correspondientes a las estrategias de evaluación Lowest, LowerLater y LowerEarlier.

A la hora de buscar las estrategias de evaluación con mejores resultados se tendrán en cuenta las estrategias de selección Best y BestRandom, ya que son las que recogen los mejores resultados de la fase de evaluación. Comparando los resultados en las tablas correspondientes a las estrategias de evaluación Lowest, LowerLater y LowerEarlier se obtiene:

- La estrategia LowerEarlier obtiene los mejores resultados para instancias pequeñas con pocos contenedores o instancias grandes con muchos contenedores.
- La estrategia Lowest obtiene buenos resultados pero nunca los mejores para ninguna instancia.
- La estrategia LowerLater obtiene los mejores resultados para todas las demás instancias.

Es lógico que las estrategias LowerLater y LowerEarlier obtengan los mejores resultados en las instancias, ya que éstas son las estrategias que más datos tienen en cuenta. Por una lado, LowerLater sigue un criterio que funciona en muchas instancias de diferente tipo, ya que mantiene las pilas igualadas en altura y con prioridad para las que contengan bloques que se extraigan más tarde. Sin embargo, si no se encuentran pilas donde el bloque colocado sea el siguiente a extraer, supondrá una futura recolocación del mismo. Por otro lado, LowerEarlier no mantiene las pilas igualadas en altura, trata de ordenar las pilas en orden de salida, lo cual beneficia a la hora de extraer bloques. Sin embargo, si no existen pilas para mantener el orden de extracción, esto supondrá la futura recolocación de los bloques en esa pila. Según el caso existen más o menos recolocaciones para una u otra estrategia, lo que supone tiempos de espera y, en definitiva, un peor resultado.

A modo de resumen de este apartado, se puede decir que:

- Las estrategias de evaluación que mejor funcionan son LowerLater y LowerEarlier, ya que junto con las estrategias de selección Best y Worst, consiguen el mejor y el peor resultado de entre todas las estrategias de evaluación. Una funciona mejor que otra según la instancia.

- Las estrategias de evaluación Lowest y Highest funcionan peor que las anteriormente mencionadas, sin embargo obtienen resultados acordes a lo esperado por estas estrategias.

- La estrategia RandomEval es la peor que funciona debido a que no obtiene ningún dato del estado de las pilas, por lo que el resultado obtenido no refleja distinción entre elegir una pila u otra.

Capítulo 5

Conclusiones y líneas de trabajo futuras

El almacenado de bloques para el suministro de los mismos a una serie de clientes es importante para la subsistencia y supervivencia, tanto de la economía local como de las personas; y es importante que seamos capaces de hacerlo de manera rápida y sencilla.

Éste es el objetivo de este Trabajo Fin de Grado; proponer estrategias e identificar las estrategias más convenientes para llevar a cabo el almacenado y suministro de bloques en un almacén, que deberán ser entregados a clientes, obteniendo los movimientos a realizar en cada instante de tiempo para minimizar la espera del cliente.

Después de estudiar varias estrategias posibles en el almacenado y suministro de bloques en un almacén, se sabe que no existe una estrategia que sea la mejor en todos los casos, sino que depende del tamaño del problema y del almacén usado.

Para concluir, se puede afirmar que los objetivos marcados al comienzo de este Trabajo de Fin de Grado han sido cumplidos. Sin embargo, el desarrollo no finaliza aquí. Aún queda trabajo por realizar para poder garantizar su eficaz funcionamiento y rendimiento. El estado final de este Trabajo puede ser el punto de partida para próximos Trabajos de Fin de Grado.

Así pues, las principales líneas de desarrollo a continuar serían las enumeradas a continuación:

- Perfeccionar las estrategias usadas para la obtención de mejores resultados.
- Combinar estrategias para lograr soluciones que no se basen en una sola estrategia y, con un estudio más exhaustivo, identificar más exactamente las instancias para las que funcionan mejor ciertas estrategias y usar cada cual según el caso.
- Hacer un balance entre el beneficio del cliente y el del almacén, optimizando tanto el coste derivado del uso de las grúas de contenedores como la espera del cliente.

Capítulo 6

Conclusions and future work

Nowadays, the block storage for supplying customers is important for subsistence and survival of the local economy and of the people; and it is important to be able to do this quickly and easily.

This is the objective of this Final Degree Project; propose a variety of strategies and identify the most suitable ones to perform the storage and delivery of blocks in a warehouse, to be delivered to customers, getting the moves to be made at each instant of time to minimize customer waiting.

After studying several possible strategies in the storage and supply of blocks in a warehouse, it is known that there is no strategy that is the best in all cases, but depends on the size of the problem and the used storage.

In conclusion, it can be said that the objectives at the beginning of this Final Degree Project have been reached. However, the development does not end here. There is still work to be done to ensure effective operation and performance. The final status of this Project could be the starting point for future Projects.

Therefore, the main lines of development for future work are listed below:

- Refine the used strategies to obtain better results.
- Combine strategies to achieve solutions that are not based on a single strategy and identify more precisely the strategies that perform better with certain instancies and use each as appropriate.
- Make a balance between the benefit of the client and the warehouse, optimizing the costs arising from the use of container cranes as much as the customer waiting.

Capítulo 7

Presupuesto

En este capítulo se especifica un presupuesto que indica cuánto costaría realizar este Trabajo de Fin de Grado si se tratase de un trabajo encargado por un cliente.

Se define una tabla con la lista de tareas realizadas en este Trabajo de Fin de Grado. Junto a cada actividad, se detalan las horas empleadas en ella y el precio por hora calculado.

El precio por hora que se considerará en este presupuesto es de 30€/hora.

Actividad	**Duración**	**Precio**
Diseño base del entorno	20 horas	600 €
Diseño del Algoritmo Heurístico	10 horas	300 €
Diseño de estrategias	15 horas	450 €
Implementación de generador de instancias	0,5 horas	15 €
Unificación de código	5 horas	150 €
Pruebas de funcionamiento	15 horas	450 €
Mejora y optimización	10 horas	300 €
Corrección de errores	10 horas	300 €
Pruebas finales	1 hora	30 €
Análisis de resultados	8 horas	240 €
Total	**94,5 horas**	**2835** €

Apéndice A

Glosario de términos

NP-duro : Un problema Np-duro o Np-complejo pertenece al conjunto de los problemas de decisión que contiene los problemas H tales que todo problema L en NP puede ser transformado polinomialmente en H. Esta clase puede ser descrita como aquella que contiene a los problemas de decisión que son como mínimo tan difíciles como un problema de NP.

Java : Java es un lenguaje de programación de propósito general orientado a objetos cuya intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web.

LIFO : El término LIFO (del inglés Last In First Out, último en entrar, primero en salir) se utiliza en estructuras de datos y teoría de colas. Guarda analogía con una pila de platos, en la que los platos van poniéndose uno sobre el otro, y si se quiere sacar uno, se saca primero el último que se ha puesto.

Bibliografía

[1] Christopher Expósito-Izquierdo, Belén Melián-Batista, and J. Marcos Moreno-Vega. A domain-specific knowledge-based heuristic for the blocks relocation problem. *Advanced Engineering Informatics*, 28(4):327 – 343, 2014.

[2] Florian Forster and Andreas Bortfeldt. A tree search procedure for the container relocation problem. *Computers & Operations Research*, 39(2):299 – 309, 2012.