Minimizando los Tiempos de Espera en las Operaciones de Retirada de Bloques en Almacenes Bidimensionales

Israel López-Plata, Christopher Expósito-Izquierdo, Eduardo Lalla-Ruiz, Belén Melián-Batista, and J. Marcos Moreno-Vega

Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38271 La Laguna, Spain
{iloppla, cexposit, elalla, mbmelian, jmmoreno}@ull.es

Resumen Este trabajo introduce el problema de minimización de tiempos de espera durante la retirada de bloques en almacenes bidimensionales. Para resolverlo se propone un algoritmo heurístico, denominado *Look Ahead Heuristic*, el cual persigue anticipar la disponibilidad del próximo bloque a entregar antes de que sea solicitado. Los resultados computacionales demuestran que el algoritmo heurístico propuesto es capaz de obtener soluciones de alta calidad en tiempos computacionales reducidos para un amplio conjunto de escenarios propuestos.

Keywords: Almacén, Recolocación, Grúa de apilado, Heurística

1. Introducción

Los almacenes son complejas infraestructuras logísticas en cadenas de suministros y dedicadas a mantener mercancías temporalmente hasta su posterior retirada. La relevancia de estas infraestructuras surge de su rol como puntos intermedios en los flujos de mercancías, desde productores hasta clientes. Un tipo de almacén de especial importancia son las terminales marítimas de contenedores dado el gran volumen de mercancías que gestionan a nivel mundial. [8] presenta una revisión exhaustiva de la gestión de cadenas de suministro.

Las mercancías en los almacenes habitualmente se encuentran consolidadas. Esto significa que se hayan empaquetadas en unidades de carga robustas, denominadas *bloques* [6]. Satisfacer la retirada oportuna de los bloques desde sus ubicaciones actuales cuando éstos son requeridos por clientes constituye, según diversos autores [4], el principal indicador de la competitividad de un almacén.

En este trabajo (i) se describe formalmente la retirada de bloques desde sus ubicaciones actuales en almacenes bidimensionales con un criterio de calidad de servicio. Con el objetivo de abordar este problema, (ii) se propone un algoritmo heurístico eficiente que permite anticipar paulatinamente la disponibilidad de los bloques solicitados por los clientes.

El resto de este trabajo se organiza tal como sigue. El problema de minimización de tiempos de espera y los principales trabajos relacionados son introducidos en la Sección 2. La Sección 3 presenta un algoritmo heurístico para resolver el problema bajo estudio. Los resultados computacionales son presentados en la Sección 4. Finalmente, la Sección 5 describe las principales conclusiones extraídas de la realización del trabajo y sugiere varias líneas de trabajo futuras.

2. Descripción del Problema

El problema de minimización de tiempos de espera en almacenes bidimensionales persigue retirar bloques homogéneos, denotados como $C = \{1, 2, \ldots, nC\}$, desde sus ubicaciones actuales. Sin pérdida de generalidad, se considera que las operaciones sobre los bloques son realizadas mediante una grúa de apilado. El objetivo del problema es minimizar la espera de los bloques durante su retirada.

El almacén bidimensional está compuesto por un conjunto de pilas, $S = \{1, 2, \ldots, nS\}$, y conjunto de alturas, $T = \{1, 2, \ldots, nT\}$. De esta manera, su capacidad está denotada como $M = nS \times nT$, medida en número de bloques. Cada posición del almacén puede almacenar un bloque durante cada instante de tiempo. En este contexto, la pila en que el bloque $c \in C$ está almacenado está denotada como $s(c) \in S$, mientras que su altura es $t(c) \in T$. Sin embargo, los bloques deben estar almacenados de acuerdo a una política de apilado, de manera que deben estar situados en el suelo o encima de otros bloques. Esto es, de acuerdo a la estrategia *último en llegar, primero en salir* [1].

Cada bloque $c \in C$ debe ser retirado durante su instante de retirada previsto, denotado como $e(c)$, el cual indica el instante de tiempo en que cierto cliente espera retirar el bloque. El tiempo de espera de c está definido como el período de tiempo entre su instante de retirada previsto y su instante de retirada real, denotado como $r(c)$ y donde $r(c) \geq e(c)$. Este tiempo de espera está denotado como $wt(c) = r(c) - e(c)$. Por tanto, la función objetivo se expresa como sigue:

$$\text{mín} \sum_{n=1}^{nC} wt(c) \tag{1}$$

El almacén bidimensional es servido por una única grúa de apilado, la cual está destinada a proporcionar y organizar los bloques. Los movimientos permitidos se describen a continuación:

- *Retirada de un bloque.* Un bloque situado en la parte alta de una pila es eliminado del almacén después de su tiempo de retirada esperado.
- *Recolocación de un bloque.* Un bloque situado en la parte alta de una pila es reubicado en la parte alta de otra pila con al menos una posición libre.

Hasta ahora, y de acuerdo al conocimiento de los autores del presente trabajo, los problemas de retirada de bloques en almacenes bidimensionales han sido únicamente tratados en la literatura con el objetivo de minimizar el número de movimientos a realizar. [7] presenta un Branch and Bound para resolver

el problema de forma exacta en el cual existen relaciones de precedencia entre bloques individuales y entre grupos de bloques. [2] propone una representación binaria del problema que es incluida dentro de una heurística eficiente para el mismo. [3] propone un algoritmo de programación dinámica que representa todos los potenciales estados del almacén. En el mismo trabajo, los autores también desarrollan una heurística que mitiga el crecimiento exponencial del número de estados. [9] propone varias estrategias para determinar la ubicación de los bloques cuando van a ser almacenados y reubicados en el almacén. Recientemente, [5] ha propuesto un Branch and Bound que incluye una estrategia inteligente para explorar sólo los nodos más prometedores del árbol, lo que hace que pueda obtener soluciones óptimas en reducidos tiempos de computación.

3. Propuesta Algorítmica

Para resolver el problema de minimización de tiempos de espera presentado en la Sección 2 del presente trabajo, se propone un algoritmo heurístico, denominado *Look Ahead Heuristic*. El razonamiento principal en el que se basa este algoritmo es en anticipar la disponibilidad del siguiente bloque a ser entregado.

En términos generales, el algoritmo intenta realizar movimientos de recolocación en aquellos instantes de tiempo en los que no se debe entregar ningún bloque, con el objetivo de tener el siguiente bloque a entregar libre lo antes posible. Con ello se intentan prever los movimientos a realizar para entregar el siguiente bloque, consiguiendo así que en el instante de tiempo en el que se debe realizar la entrega el número de movimientos necesario para liberar el bloque sea mínimo y, de esta forma, minimizar el tiempo de espera del cliente.

El pseudo-código de esta heurística se ilustra en el Algoritmo 1. En él hay que destacar los siguientes elementos: $b_{entrega}$ (siguiente bloque a entregar) y H (horizonte de planificación). Como primer paso, se obtiene el primer bloque a entregar (línea 1). Una vez este bloque ha sido identificado, se recorre el horizonte de planificación para definir los movimientos a realizar en cada instante del mismo. Si el instante actual es superior o igual al instante esperado de entrega del bloque a suministrar y el bloque se encuentra disponible para su entrega (línea 3), entonces se procede a la entrega del bloque (línea 4) y a identificar un nuevo bloque objetivo para liberar (línea 5). Se considera que un bloque $c \in C$ se encuentra disponible para entregar si el conjunto de bloques que se encuentran apilados encima del mismo, $O(c)$, está vacío. Esto se define tal como sigue:

$$O(c) = \{c' \in C \mid (s(c') = s(c)) \wedge (t(c') > t(c))\}. \tag{2}$$

En caso contrario, y sólo si el bloque no está liberado para su entrega (línea 7), se realiza la recolocación de los bloques que se encuentren encima de éste (línea 9), con el objetivo de dejar al bloque disponible. El bloque que se encuentra en la parte superior de una pila se denota como c_s y se define tal como sigue:

$$c_s = \{c \in C \mid (s(c) = s) \wedge (\nexists c' \in C : s(c') = s \wedge t(c') > t(c))\}. \tag{3}$$

Algorithm 1 Pseudocódigo de la heurística utilizada para resolver el problema de minimización de tiempos de espera en almacenes bidimensionales

```
1: b_entrega ← Primer bloque a entregar
2: for (i_actual ← 1 ... H) do
3:    if (i_actual ≥ e(b_entrega) y O(b_entrega) = ∅) then
4:       Entregar el bloque b_entrega
5:       b_entrega ← Obtiene el siguiente bloque a entregar
6:    else
7:       if (O(b_entrega) ≠ ∅) then
8:          s ← Pila de b_entrega (i.e., s(b_entrega))
9:          Recolocar c_s
10:      end if
11:   end if
12: end for
```

En la Figura 1 se ilustra el funcionamiento del algoritmo propuesto para la entrega de 2 bloques. Como se puede observar, el primer paso es identificar el primer bloque a entregar, en este caso es $b_{entrega} = 1$, el cual debe entregarse en el instante 2. En el instante 1 se realiza la recolocación del bloque 6 desde la pila 2 a la 1, liberando así al bloque que se desea entregar. Con ello se consigue que justo en el instante 2 el bloque 1 pueda ser entregado, consiguiendo gracias al movimiento predictivo del instante 1 que el tiempo de espera en el momento de la entrega sea mínimo. Una vez se ha suministrado el bloque 1 se identifica el siguiente, en este caso el número 2, que debe entregarse en el instante 5.

Al igual que en el caso anterior, se aprovecha el siguiente instante de tiempo ($i_{actual} = 3$) para recolocar el bloque 11, el cual se encuentra encima del bloque 2, dejándolo así libre. Debido a que el bloque 2 ya se encuentra libre y aún no ha llegado su momento de entrega, en el instante 4 no se realiza ningún movimiento y no es hasta el instante 5 hasta que no se realiza la entrega del mismo.

3.1. Estrategias de Recolocación

Un factor que posee una influencia relevante en la calidad de los resultados devueltos por *Look Ahead Heuristic* es el movimiento de recolocación que se realiza en caso de que sea necesario, el cual traslada un bloque de la pila que se desea liberar a otra de su elección. Una buena estrategia de recolocación permite liberar bloques para su entrega en un menor número de movimientos, lo que implica la reducción de los tiempos de espera para la entrega de dichos bloques. Es por ello por lo que en este trabajo se estudia el comportamiento de *Look Ahead Heuristic* con diversas estrategias de recolocación existentes en la literatura. Éstas son:

1. *Random*. La pila de destino de la recolocación es seleccionada aleatoriamente entre aquellas que tengan algún hueco disponible, denotado como $\mathcal{S}$.
2. *Conflict Minimization (Min)*. La pila de destino se elige de forma aleatoria entre todas aquellas que tengan algún hueco disponible y cuyos bloques ten-

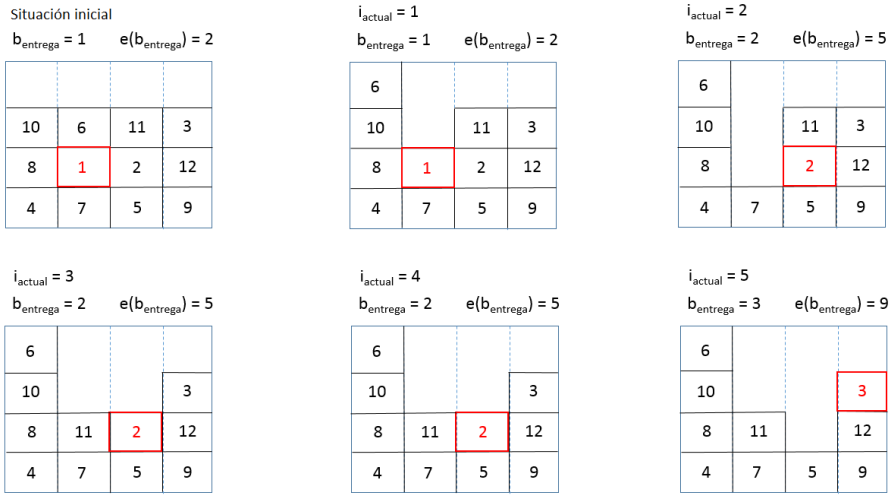


Figura 1: Ejemplo de ejecución del algoritmo en la entrega de los 2 primeros bloques

gan tiempo de entrega posterior al del bloque a recolocar. El conjunto de pilas que cumplen estas condiciones se define como sigue:

$$\mathcal{X}(c) = \{s \in \mathcal{S} \mid e(c) < e(min(s))\}, \tag{4}$$

donde $min(s)$ es:

$$min(s) = \{c \in C \mid (s(c) = s) \wedge (\nexists c' \in C : s(c') = s \wedge e(c') < e(c))\} \tag{5}$$

En el caso de que el conjunto $\mathcal{X}(c)$ sea vacío, se realiza una recolocación aleatoria.

3. *Flexibility Optimization (Flex).* La pila de destino es aquella que tiene algún espacio disponible y que obtiene el menor de los valores de acuerdo con la siguiente función de evaluación:

$$f_{Flex}(c,s) = \begin{cases} 1 + \text{máx}_{i \in C}\{e(i)\}, & B(s) = \emptyset \\ min(s), & (B(s) \neq \emptyset) \wedge (e(c) < min(s)) \\ 2 \cdot (1 + \text{máx}_{i \in C}\{e(i)\}) - min(s), & (B(s) \neq \emptyset) \wedge (e(c) > min(s)) \end{cases}, \tag{6}$$

donde $\text{máx}_{i \in C}\{e(i)\}$ representa el mayor de los tiempos de entrega de los bloques y $B(s)$ es el conjunto de bloques que se encuentran en una pila:

$$B(s) = \{c' \in C \mid s(c') = s\}. \tag{7}$$

De entre todas las pilas que cumplan con las condiciones y que tengan la menor puntuación, se escoge una de forma aleatoria.

4. *Parameterized Flexibility Optimization (PFO).* Es una extensión de la estrategia anterior, en la cual se escoge una pila de destino entre aquellas con menor puntuación con una probabilidad de $1-\rho$, mientras que se escoge una pila entre aquellas con segunda mejor puntuación con una probabilidad de ρ. Así pues, los pasos a seguir por esta estrategia son los siguientes:
   a) Se define $\mathcal{Y} = \{s \in \mathcal{S} \mid f_{Flex}(c,s) \leq f_{Flex}(c,s'), \forall s' \in \mathcal{S}\}$.
   b) Se elige una pila s_1 del conjunto $\mathcal{Y}$ de forma aleatoria.
   c) Se define $\mathcal{S}' = \mathcal{S} - \mathcal{Y}$ y $\mathcal{Z} = \{s \in \mathcal{S}' \mid f_{Flex}(c,s) \leq f_{Flex}(c,s'), \forall s' \in \mathcal{S}'\}$.
   d) Se elige una pila s_2 del conjunto $\mathcal{Z}$ de forma aleatoria.
   Así pues, s_1 se selecciona con una probabilidad de $1-\rho$ y s_2 con una probabilidad de ρ.

4. Resultados Computacionales

Esta sección tiene como objetivo evaluar el rendimiento del algoritmo propuesto en la Sección 3 para resolver el problema de minimización de tiempos de espera en almacenes bidimensionales. La técnica de optimización se ha implementado en Java Standard Edition 7, mientras que las ejecuciones se han realizado en un ordenador equipado con un Intel i7-3.50 GHz y 8 GBs de RAM.

Los experimentos computacionales se han llevado a cabo sobre un conjunto de instancias realistas de diferentes tamaños (*i.e.*, $nT \times nS$). Los tamaños de instancias disponibles son los siguientes: 3x3, 3x4, 3x5, 3x6, 3x7, 3x8, 4x4, 4x5, 4x6, 4x7, 5x4, 5x5, 5x6, 5x7, 5x8, 5x9, 5x10, 6x6, 6x10, 10x6 y 10x10. En cada caso, se incluyen dos alturas libres de bloques en la parte superior de los almacenes para poder realizar las recolocaciones. Finalmente, por cada uno de los diferentes tamaños se dispone de un conjunto de 40 instancias diferentes.

Para cada una de las ejecuciones realizadas se define el parámetro δ. Éste representa la separación del tiempo de entrega previsto entre dos bloques consecutivos en prioridad de entrega. Así pues, con $\delta = 1$ los bloques esperan ser entregados sin esperas entre ellos, mientras que, por ejemplo, con $\delta = 5$, la entrega de los bloques está separada por 5 instantes de tiempo. Este parámetro tiene gran influencia en el comportamiento del algoritmo desarrollado así como en los resultados obtenidos con el mismo, ya que un δ mayor permite un mayor margen de movimientos para poder entregar un bloque sin demora. Para poder conocer cómo influye la separación de los tiempos de entrega, se han realizado experimentaciones con diferentes valores de δ, éstos son 1, 3, 5 y 10.

4.1. Evaluación de Rendimiento

La experimentación realizada se ha basado en realizar 10 ejecuciones para cada una de las combinaciones resultantes de variar los siguientes factores:

- Instancias
- Valores de δ
- Estrategias de recolocación explicadas en la Sección 3

Los resultados de las ejecuciones se han agrupado en base a las dimensiones de las instancias, representando cada grupo el valor medio de 400 ejecuciones (40 instancias y 10 ejecuciones por instancia). En el Cuadro 1 se muestran, para cada grupo de instancias, los resultados obtenidos en términos de valor de función objetivo y tiempo (en nanosegundos) de la heurística propuesta con cada una de las estrategias de recolocación con $\delta = 1$. Para las instancias de menor tamaño (3x3 y 3x4) se muestran también los resultados de ejecución del modelo matemático ejecutado con CPLEX 12.6 [1], que proporciona el valor objetivo óptimo y el tiempo de ejecución del mismo (en segundos). No se incluyen los resultados para instancias de mayor tamaño ya que CPLEX no es capaz de obtener soluciones óptimas en tiempos de ejecución razonables.

De los resultados con $\delta = 1$ se puede deducir que el algoritmo heurístico propuesto permite resolver el problema de minimización de tiempos de espera en almacenes bidimensionales de forma eficiente. En concreto, proporciona soluciones de alta calidad dada su cercanía al valor óptimo mediante tiempos de computación reducidos, menos de 1 segundo. Además de esto, la heurística es capaz de obtener soluciones factibles para instancias de mayor tamaño, algo de lo que es incapaz el modelo matemático.

Al analizar las estrategias, se puede ver que, cuanto mayor es la dimensión del problema, mejores resultados se obtienen, ya que la recolocación se encuentra más elaborada. Así pues, la estrategia que mejores resultados devuelve es la *Parameterized Flexibility Optimization*, mientras que la peor es la *Random*.

Por otro lado, la separación entre los tiempos de entrega de los bloques repercute directamente en los resultados. Para poder analizar esta influencia, se ha realizado una comparación entre los resultados de la heurística con la estrategia *Parameterized Flexibility Optimization* para diferentes valores de δ.

En la Figura 2a se aprecia cómo influye el valor de δ en las instancias de pequeño tamaño. Como se puede apreciar, para $\delta = 1$ sí existen tiempos de espera, mientras que si se aumenta el tamaño de la separación los tiempos de espera se reducen a casi 0. El caso de $\delta = 10$ no se muestra en la tabla ya que en todos los casos el valor de la función objetivo es 0.

En las instancias de tamaño grande (Figura 2b) se observa la misma tendencia que para las instancias de dimensiones pequeñas, con unos valores de función objetivo mucho mayores para $\delta = 1$ que para el resto. Además, se puede ver que según aumenta el tamaño de las instancias los tiempos de espera son mayores, incluido en las instancias con δ mayor a 1.

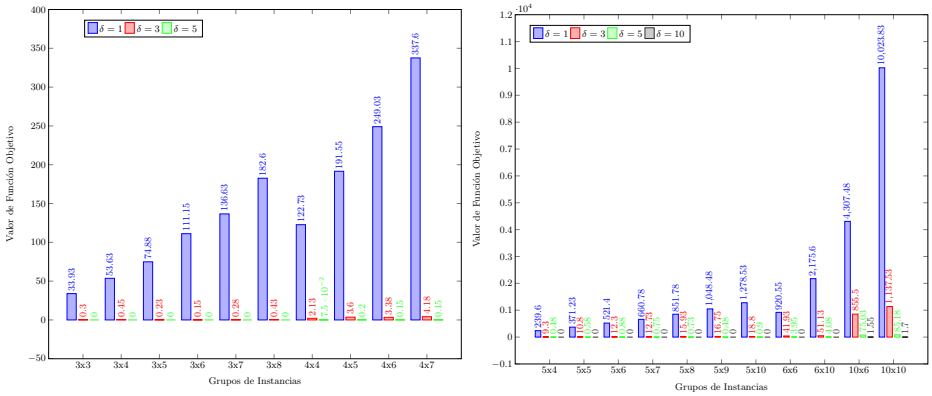
De esto se puede deducir la existencia de una relación directa entre el tamaño de las instancias, la separación de los tiempos de entrega de cada uno de los bloques y el valor de función objetivo. Además, se aprecia una diferencia sustancial en valores de función objetivo entre las ejecuciones con $\delta = 1$, la cual no permite aprovechar la naturaleza predictiva del algoritmo propuesto, y el resto, siendo éstas últimas soluciones de mucha mejor calidad.

Este análisis demuestra que el algoritmo es capaz de aprovechar los tiempos sin entregas para liberar el siguiente bloque a entregar, obteniendo mejores re-

[1] http://www.ibm.com/software/products/es/ibmilogcpleoptistud/

nT	nS	f_{opt}	t_{opt} (s.)	f_{Random}	t_{Random} (ns.)	f_{Min}	t_{Min} (ns.)	f_{Flex}	t_{Flex} (ns.)	f_{PFO}	t_{PFO} (ns.)
3	3	33.825	669.376	42.890	10415.552	37.705	6423.332	33.925	7361.047	43.015	8354.255
	4	51.889	31839.200	77.132	12302.557	59.432	8230.210	53.625	10198.507	64.982	11216.335
	5	-	-	112.847	14020.680	82.960	10671.030	74.875	13842.127	88.160	14911.230
	6	-	-	167.817	15456.535	122.580	12703.477	111.150	16785.420	125.320	17975.375
	7	-	-	213.335	19251.900	148.977	16843.157	136.625	22166.345	149.285	23125.427
	8	-	-	295.557	26276.155	201.320	20473.975	182.600	37385.572	199.550	35551.987
4	4	-	-	175.587	15405.177	139.957	13189.907	122.725	16100.125	149.975	19164.257
	5	-	-	287.355	25711.650	221.185	19017.795	191.550	24608.312	223.282	31710.597
	6	-	-	393.300	23912.292	290.450	22727.690	249.025	27436.187	281.015	30405.177
	7	-	-	553.802	27058.710	389.960	28171.745	337.600	34953.305	377.185	39479.995
5	4	-	-	344.077	23617.210	289.385	23172.447	239.600	24473.597	299.645	29563.767
	5	-	-	568.110	26415.157	461.240	27362.417	371.225	38584.040	445.722	49260.485
	6	-	-	839.937	41655.660	645.680	35840.695	521.400	48296.120	606.570	57701.275
	7	-	-	1113.062	37321.425	825.845	44343.500	660.775	57399.755	757.032	59462.137
	8	-	-	1471.492	48123.922	1070.212	43035.940	851.775	58651.745	955.152	69062.990
	9	-	-	1854.210	52327.770	1316.137	52227.282	1048.475	76562.977	1166.627	87238.302
	10	-	-	2306.845	59241.915	1586.835	83176.682	1278.525	89374.445	1400.012	102709.835
6	6	-	-	1472.742	42419.057	1194.800	49335.315	920.550	65321.042	1076.525	68165.967
	10	-	-	4010.345	78602.862	2965.655	96730.092	2175.600	124195.165	2443.417	138600.682
10	6	-	-	6811.447	88902.967	6223.242	116192.672	4307.475	138040.455	5222.390	161396.827
	10	-	-	18793.420	177086.415	16308.965	238773.442	10023.825	311570.897	11567.177	326915.182

Cuadro 1: Resultados obtenidos sobre el conjunto de instancias cuando $\delta = 1$



(a) Instancias pequeñas (b) Instancias grandes

Figura 2: Comparación de resultados según el valor del parámetro δ

sultados cuanto más instantes de tiempo se disponen para esta operación antes de la entrega.

5. Conclusiones y Líneas de Trabajo Futuras

En este trabajo se presenta el problema de minimización de tiempos de espera en almacenes bidimensionales. Para resolverlo, se propone un algoritmo heurístico llamado *Look Ahead Heuristic* que está basado en la anticipación de la disponibilidad del siguiente bloque a entregar.

Para evaluar el rendimiento de la heurística se realiza una experimentación sobre un gran número de escenarios, los cuales varían en tamaño y en separación entre los tiempos de entrega (mediante el parámetro δ). Para cada uno de estos escenarios se ha evaluado la ejecución de la heurística con las diferentes estrategias de recolocación de bloques encontradas en la literatura, tanto en valor de la función objetivo como en tiempo de ejecución. Además, para algunos conjuntos de instancias de tamaño pequeño se ha podido evaluar la calidad de las soluciones respecto a las óptimas proporcionadas por CPLEX.

Los resultados de la experimentación demuestran que la heurística propuesta es capaz de proporcionar resultados de alta calidad en un tiempo de computación pequeño, lo que permite abordar escenarios de gran tamaño. La comparación de las diferentes estrategias de recolocación aplicadas a la heurística propuesta demuestra que la mejor es la *Parameterized Flexibility Optimization*, ya que en la mayoría de los casos proporciona soluciones con los menores tiempos de espera. Por otro lado, el análisis de los resultados para distintos valores de δ demuestra que a mayor separación en la entrega de bloques, menores son los tiempos de espera. De ello se puede deducir que la heurística propuesta aprovecha oportunamente los instantes sin entrega para anticipar la disponibilidad del siguiente bloque a retirar.

Teniendo en cuenta los resultados obtenidos, se puede concluir que la *Look Ahead Heuristic* combinada con una estrategia de recolocación *Parameterized Flexibility Optimization* es capaz de resolver el problema planteado, obteniendo soluciones de alta calidad en tiempos de computación reducidos.

Como líneas de trabajo futuras, se propone la integración del problema en un sistema de toma de decisiones en el cual intervengan los vehículos de transporte horizontal, operarios y múltiples almacenes. Por su parte, en futuros trabajos se continuará el desarrollo del algoritmo aquí propuesto para que se consideren varios bloques a liberar antes de su solicitud.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad de España (project TIN2012-32608). Christopher Expósito-Izquierdo y Eduardo Lalla-Ruiz agradecen al Gobierno de Canarias el apoyo financiero que reciben a través de sus becas de doctorado.

Referencias

1. Marco Caserta, Silvia. Schwarze, and Stefan. Voβ. Container rehandling at maritime container terminals. In J.W. Böse, R. Sharda, and S. Voβ, editors, *Handbook of Terminal Planning*, volume 49 of *Operations Research/Computer Science Interfaces Series*, pages 247–269. Springer New York, 2011.
2. Marco Caserta, Silvia Schwarze, and Stefan Voβß. A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In Carlos Cotta and Peter Cowling, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 5482 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin / Heidelberg, 2009.
3. Marco Caserta, Stefan Voβ, and Moshe Sniedovich. Applying the corridor method to a blocks relocation problem. *OR Spectrum*, 33(4):915–929, 2011.
4. Felix T.S. Chan and H.K. Chan. Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. *Expert Systems with Applications*, 38(3):2686 – 2700, 2011.
5. Christopher Expósito-Izquierdo, Belén Melián-Batista, and J. Marcos Moreno-Vega. An exact approach for the blocks relocation problem. *Expert Systems with Applications*, 42(17–18):6408 – 6422, 2015.
6. Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539 – 549, 2010.
7. Kap Hwan Kim and Gyu-Pyo Hong. A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940 – 954, 2006.
8. M.T. Melo, S. Nickel, and F. Saldanha da Gama. Facility location and supply chain management - A review. *European Journal of Operational Research*, 196(2):401 – 412, 2009.
9. Rui Jorge Rei and João Pedro Pedroso. Heuristic search for the stacking problem. *International Transactions in Operational Research*, 19(3):379–395, 2012.