Búsqueda de Vecindad Variable para el problema de la agrupación y recogida de pedidos online en almacenes logísticos

Sergio Gil-Borrás, Abraham Duarte, Antonio Alonso-Ayuso y Eduardo G. Pardo

***Resumen*—En este trabajo se presenta la adaptación de un algoritmo de Búsqueda de Vecindad Variable Básica al problema de agrupación y recogida de pedidos *online* en almacenes logísticos. La recogida de pedidos en almacenes se ha convertido, en la actualidad, en una importante tarea operacional como parte de la cadena de suministro. En particular, debido a los costes derivados de esta actividad se ha tornado necesario tratar de optimizar el proceso de recogida, de modo que los productos asociados a los pedidos sean entregados de forma eficiente. Existen diferentes políticas de recogida, entre las que destacan: (i) la recogida de pedidos directa, es decir, cada pedido que llega al almacén es recogido directamente por un trabajador en orden de llegada; y (ii) la recogida por lotes, en la que varios pedidos son agrupados en un mismo lote y asignados a un trabajador, para ser recogidos simultáneamente. En este trabajo se aborda una variante del problema de recogida de pedidos, basada en la política de agrupación en lotes. En concreto, la variante abordada tiene en consideración que los pedidos llegan *online* al almacén, es decir, que no están todos disponibles al comienzo del turno de recogida, sino que van llegando a medida que la jornada de trabajo avanza. La función objetivo consiste en minimizar el tiempo máximo que un pedido permanece en el sistema.**

***Palabras clave*—** Búsqueda de Vecindad Variable, Agrupación en lotes, Heurísticas, *Online Order Batching Problem*.

I. Introducción

En los últimos años las compras a través de Internet se han popularizado en la sociedad. Derivado de esto, ha surgido la necesidad de mejorar los procesos que se producen dentro de los almacenes logísticos, para llevar a cabo una rápida y eficaz gestión de los pedidos, así como una reducción de costes de procesamiento de los mismos. Esto ha motivado el estudio de diferentes problemas que se producen dentro de este contexto, tan demandado por empresas de distribución. Entre los problemas que surgen, se pueden encontrar, entre otros, problemas relacionados con la recepción de mercancías, su almacenamiento y la recogida de las mismas cuando se recibe un pedido. En este último contexto, es donde se enmarca el presente trabajo. En concreto, se estudia cómo hacer eficiente la recogida de pedidos *online* que llegan al almacén. Existen diferentes variantes de este problema, en función de factores tales como: el número de personas recogiendo los pedidos simultáneamente; si los pedidos tienen o no fecha límite de entrega; e incluso en función de la estructura concreta de los almacenes considerados (varios bloques de estanterías, número de pasillos paralelos, existencia de pasillos diagonales, etc.).

El problema que se aborda aquí es el denominado "*Online Order Batching Problem*" (OOBP). Este problema consiste en determinar el orden en el que se recogen los pedidos que van llegando al almacén de manera constante, así como la ruta necesaria para recogerlos. Para ello, se conforman una serie de lotes de recogida, formados por un conjunto de pedidos, con un tamaño máximo por lote. En este caso, cada uno de los lotes será recogido por un único trabajador. El objetivo consiste en minimizar el tiempo máximo que transcurre desde que un pedido entra en el almacén hasta que este ha sido procesado para su envío. Este tiempo es comúnmente denominado "*makespan*" en la literatura asociada a los procesos industriales. El tiempo en el que un pedido está en el almacén es, en realidad, la suma de dos tiempos. Por un lado, el tiempo de procesado del lote (compuesto a su vez por lo que tarda en generarse el lote y por el tiempo de espera hasta que este lote pase a ser recogido). Y, por otro lado, el tiempo que se tarda en recoger todo el lote por los pasillos del almacén y llevarse a la zona de entrega. Para este último proceso, la recogida de artículos, se emplean algoritmos que establecen una ruta dentro del almacén. En concreto, en este trabajo se han repasado tres de los algoritmos heurísticos más conocidos en la literatura, como son S-Shape [1], Largest-Gap [2] y Combined [3] [4], que se repasarán detalladamente en la Sección III.

En este trabajo se han empleado almacenes con dos pasillos trasversales, uno superior y otro inferior, y un conjunto variable de pasillos paralelos que los unen, donde se colocan los productos en columnas de igual tamaño, a ambos lados de cada pasillo paralelo. El número de pasillos y las dimensiones de la sala, así como de las estanterías, varía según la instancia seleccionada. En la Fig. 1 se puede ver un ejemplo de este tipo de almacén con dos pasillos transversales y cinco pasillos paralelos. Cada pasillo paralelo tiene nueve posiciones de recogida a cada uno de los lados, representadas por pequeños

Sergio Gil-Borrás, Universidad Politécnica de Madrid. Ctra. Valencia, Km. 7, 28031, Madrid, España, (e-mail: sergio.gil.borras@alumnos.upm.es).
Abraham Duarte, Universidad Rey Juan Carlos. C/Tulipán s/n, 28933, Madrid, España, (e-mail: abraham.duarte@urjc.es).
Antonio Alonso-Ayuso, Universidad Rey Juan Carlos. C/Tulipán s/n, 28933, Madrid, España, (e-mail: antonio.alonso@urjc.es).
Eduardo G. Pardo, Universidad Politécnica de Madrid. Ctra. Valencia, Km. 7, 28031, Madrid, España, (e-mail: eduardo.pardo@upm.es).

cuadrados. El almacén mostrado tiene, por lo tanto, un total de 90 posiciones de recogida. Se considera que en cada posición de recogida se almacena únicamente un posible producto. No se consideran, por lo tanto, múltiples alturas en cada posición de recogida. En el pasillo transversal inferior se sitúa, además, el depósito, que será el punto de partida para las rutas de recogida, así como el punto de entrega de los productos recogidos en cada ruta. Este depósito aparece posicionado en el centro del pasillo, si bien existen instancias en las que el depósito se encuentra en uno de los extremos del mismo.

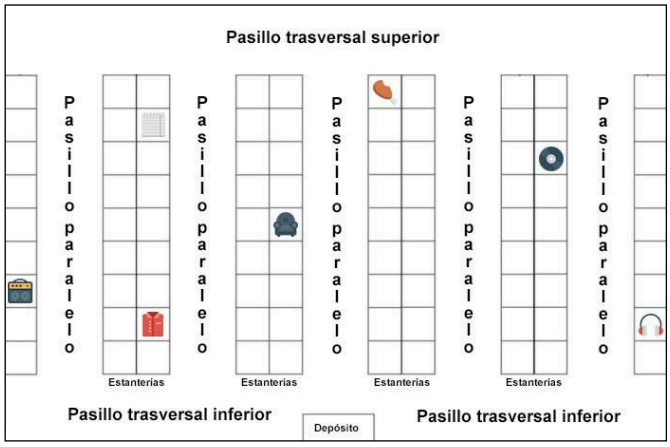

Fig. 1 – Estructura del almacén

El resto del artículo está estructurado de la siguiente forma: en la Sección II se repasa el estado del arte del problema abordado. En la Sección III se describen los algoritmos de enrutamiento más relevantes de la literatura. En la Sección IV se presentan los algoritmos heurísticos de agrupación propuestos, basados en la metodología de Búsqueda de Vecindad Variable. En la Sección V se muestran algunos resultados preliminares obtenidos y los conjuntos de instancias empleados. Por último, en la Sección VI se exponen las conclusiones de este trabajo.

II. Estado del arte

Entre los últimos artículos publicados que se recogen en la literatura, relativos a esta familia de problemas de optimización, destaca el artículo de S. Henn en 2012 [5], donde los autores proponen un algoritmo basado en la metaheurística "*Iterated Local Search*" (ILS) [6] para abordar el problema. Este algoritmo está basado en el principio de búsqueda local y, en este artículo, se compara el algoritmo propuesto con las técnicas heurísticas clásicas de "*First Come, First Served*" (FCFS) y el método de "Clarke and Wright" (C&W) [7].

Más recientemente, en el año 2015, Ricardo Pérez-Rodríguez publicó un artículo [8] resolviendo también esta variante del problema. En este artículo, los autores proponen un algoritmo basado en la estimación continua de la distribución (denominado OBCEDA por sus siglas en inglés) que a su vez está basado en los bien conocidos Algoritmos de la Estimación de la Distribución (EDA) [9]. En este trabajo, se compara la propuesta realizada con un algoritmo genético y con un algoritmo basado en la metaheurística "Tabu-Search" [10]. En este caso también se usan las mismas instancias que usa Sebastián Henn en su artículo del 2012, si bien no se realiza una comparación directa entre ambos trabajos.

Sobre otras variantes del problema, íntimamente relacionadas, destaca la publicación en estos últimos años del trabajo presentado por Rubrico en [11] donde la recogida de los pedidos se hace por múltiples trabajadores (habitualmente denominados *pickers*) en lugar de únicamente por uno, tal y como se realiza en el problema tratado en este documento. En este caso, Rubrico propone dos variaciones de un "*Incremental static scheduling scheme algorithm*", que los autores denominan "*Steepest descent insertion*" y "*On multistage rescheduling*". En este caso, los autores utilizan un conjunto de instancias generadas por ellos mismos, y se compara con el "G66 *Listing algorithm*", el cual es una extensión del algoritmo "*Graham's list scheduling algorithm*" [12]. También se realiza una comparación con el denominado "*Aspnes'93 algorithm*" [13].

Otra variante interesante del problema, publicada en 2016 [14], consiste en añadir una restricción de fecha/hora límite de entrega a cada pedido y, además, la recogida de los pedidos también se realiza con múltiples *pickers*. El algoritmo que los autores proponen es un algoritmo basado en reglas para preprocesar los pedidos antes de usar el algoritmo de agrupamiento de pedidos en lotes. Con esta estrategia, los autores consiguieron mejorar los resultados. En su caso, los algoritmos que usan para compararse son los "*Seed algorithms*" [15] y el "*Recalculation saving algorithms*" [16]. Las instancias utilizadas en este artículo siguen siendo las mismas instancias que introdujo Sebastián Henn en [5].

III. Enrutamiento

Los algoritmos de enrutamiento son los encargados de decidir el camino que seguirá el trabajador por el almacén para recoger todos los pedidos incluidos en un mismo lote. El camino comenzará en el depósito que está marcado en las siguientes figuras como "D" (del término en inglés *depot*). El camino finaliza en el mismo depósito, una vez que se han recogido todos los elementos de un mismo lote. Al igual que otros problemas de enrutamiento, este problema de optimización ha sido ampliamente trabajado con anterioridad, existiendo en la literatura algoritmos heurísticos, metaheurísticos y exactos para esta variante concreta. No obstante, los algoritmos de enrutamiento elegidos para este trabajo, y que se repasan a continuación, son algoritmos heurísticos. Estos algoritmos, aunque generan soluciones con recorridos ligeramente más largos que los algoritmos exactos o que los algoritmos metaheurísticos, son más rápidos a la hora de computar una solución. Además, en términos generales, el recorrido que ofrecen suele ser de alta calidad y más comprensible por parte de los empleados del almacén.

En la Fig. 2 se puede ver una ilustración de la estrategia de enrutamiento denominada S-Shape, también conocida en algunas referencias como estrategia Trasversal. Esta estrategia consiste en recorrer completamente todos los pasillos paralelos que contengan ítems que recoger. A excepción del último

pasillo con ítems, que dependerá de la posición del recogedor cuando deba entrar en dicho pasillo. En el caso que se acceda a él desde el pasillo trasversal inferior, entonces se recorrerá el pasillo únicamente hasta llegar al último ítem a recoger del pasillo y luego, el recogedor dará media vuelta para recorrer ese mismo pasillo de vuelta hasta llegar de nuevo al pasillo trasversal inferior y, posteriormente, volver al depósito para entregar las mercancías. No obstante, en el caso de que el *picker* se encuentre en el pasillo transversal superior, en el momento de entrar en el último pasillo, entonces este se recorrerá completamente, volviendo al pasillo transversal inferior, que contiene el depósito, para la entrega de mercancías. En la Fig. 2, están representados con color compacto negro los puntos de recogida de los ítems de un mismo lote.

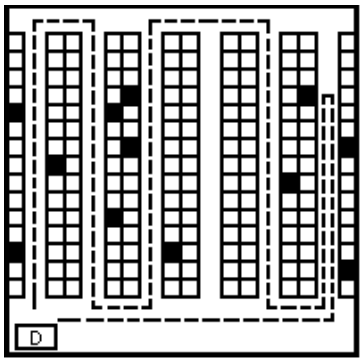

Fig. 2 – Recorrido basado en S-shape[1]

Análogamente, en la Fig. 2, se puede ver la estrategia de enrutamiento denominada Largest-Gap. En esta estrategia solo se recorrerán completamente el primer y último pasillo paralelo con ítems. El resto de pasillos paralelos solo se recorrerán, en el caso que tenga ítems a recoger. Además, el tramo de pasillo a recorrer será el comprendido entre el pasillo transversal del que se parta (inferior o superior según el caso) y el denominado "*largest-gap*". Este término se utiliza para referirse al espacio más grande que hay entre cada par de ítems consecutivos en un mismo pasillo. Una vez alcanzado el *largest-gap*, el operario encargado de la recogida debe dar media vuelta sobre sus pasos y volver al pasillo transversal del que partía. En el caso de que quedasen ítems por recoger (que se encuentren más allá del *largest-gap*) estos serán recogidos en la ruta de vuelta desde el pasillo transversal contrario. Por lo tanto, desde el pasillo trasversal superior se accederá solo a recoger los ítems que se encuentran en la zona situada antes del *largest-gap*, en la mitad superior de los pasillos paralelos. De manera similar, desde el pasillo trasversal inferior solo se recogerán los ítems que se encuentran entre el pasillo transversal inferior y la última posición de un ítem dentro del pasillo paralelo, antes del denominado *largest-gap*.

Por último, en la Fig. 3, se puede ver la estrategia denominada Combined. En esta estrategia se combinan las técnicas de las dos estrategias anteriormente explicadas.

1. La Fig. 2, Fig. 3 y Fig. 4 han sido obtenidas de la página web de la Universidad de Rotterdam: https://www.erim.eur.nl/material-handling-forum/research-education/tools/calc-order-picking-time/what-to-do/routing-strategies/, el día 13 Julio de 2018.

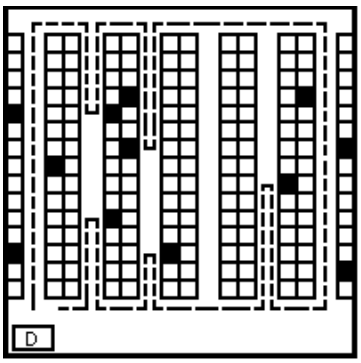

Fig. 3 – Recorrido basado en Largest Gap[1]

En la estrategia Combined, para recorrer cada pasillo paralelo se evalúa qué recorrido es más corto de hacer para recoger todos los ítems de un mismo pasillo, si hacer una estrategia S-Shape o si hacer una estrategia Largest-Gap. De este modo, a medida que se va avanzando se toma la mejor decisión para cada pasillo. De manera global, se puede afirmar que los resultados obtenidos con esta estrategia son mejores que aquellos obtenidos aplicando las anteriores estrategias expuestas de manera independiente. En la Fig. 4, se presenta un ejemplo de un recorrido empleando la estrategia Combined, para la misma instancia representada en la Fig. 2 y en la Fig. 3.

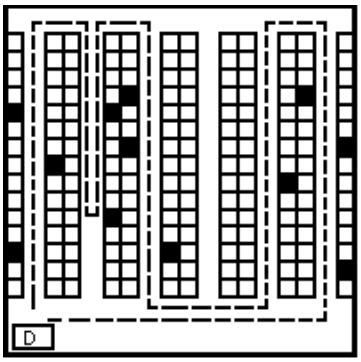

Fig. 4 – Recorrido basado en Combined[1]

IV. Agrupamiento

Para abordar el problema del agrupamiento de pedidos en lotes, se propone la combinación de dos algoritmos: un algoritmo constructivo voraz y una metaheurística, como método de mejora de la solución inicial formada por el constructivo.

El método constructivo ordena los pedidos en función del orden de llegada y parte de un lote vacío. A continuación, toma el primer pedido según el orden establecido y trata de añadirlo al último lote disponible (en el caso inicial, el lote vacío). Cuando este lote no tiene espacio suficiente crea un nuevo lote. Este proceso se repite hasta que todos los pedidos han sido asignados a un lote. En lo sucesivo se denominará a este método constructivo como *First-Come-First-Serve* (FCFS). Se propone también una variante de este método consistente en tratar de añadir el siguiente pedido en el orden establecido a cada uno de los lotes previos (no solo en el último) intentando, de ese modo, completar el espacio vacío en los lotes. A este método se le denominará en lo sucesivo como *First-Come-First-Serve* Completo (FCFS-Completo).

Además de los métodos constructivos anteriormente descritos, se propone la utilización de un método basado en la metaheurística Búsqueda de Vecindad Variable (VNS) [17]. VNS es un algoritmo metaheurístico que ayuda a resolver problemas complejos tanto de optimización combinatoria como de optimización global. En concreto, partiendo de una solución inicial, intenta mejorarla visitando distintos vecindarios. Mediante la utilización de una búsqueda local explora, en cada vecindario, las posibles soluciones prometedoras, obteniendo un óptimo local en cada uno de ellos.

Existen diferentes variantes dentro de la metodología VNS. En este trabajo, se ha escogido la variante denominada "Búsqueda de Vecindad Variable Básica" (BVNS, del inglés *Basic Variable Neighborhood Search*). El algoritmo BVNS se basa en tres métodos: un método consistente en una búsqueda local para alcanzar un óptimo local en un vecindario dado; un método de perturbación de la solución y un método que determina si debe o no cambiarse la vecindad a explorar. En Algoritmo I se presenta el pseudocódigo del método BVNS.

ALGORITMO I
FUNCIÓN BVNS

```
Función BVNS (x, k_max, t_max);
 1: repeat
 2:    k ← 1;
 3:    repeat
 4:       x' ←Shake(x, k) /* Perturbación */;
 5:       x" ← PrimeraMejora (x') /* Búsqueda local */;
 6:       x← CambioVecindad (x, x", k)
          /* Cambio de vecindad */;
 7:    until k = k_max ;
 8:    t ←CpuTime()
 9: until t > t_max ;
```

Cada iteración del método BVNS repite la ejecución de los tres métodos anteriormente mencionados, mientras que no se alcance el valor máximo de *k* (denotado por el parámetro *k_max*), que determina el máximo número de vecindades a explorar. Una vez alcanzado el valor de *k_max*, se permite realizar una nueva exploración completa si el tiempo máximo de ejecución (*t_max*) no se ha superado.

El método de búsqueda local propuesto dentro del algoritmo BVNS se presenta en el pseudocódigo de la función PrimeraMejora (ver Algoritmo II). Este método, dada una solución de partida *x* explora las soluciones en el vecindario de *x,* denotado como N(*x*), hasta que encuentra una solución de mejor calidad, en cuyo caso, utiliza esta nueva solución como punto de partida para la siguiente iteración del algoritmo. El procedimiento se repite hasta que toda la vecindad ha sido explorada. La vecindad, en el caso de esta búsqueda local, está definida por movimientos de inserción factibles, es decir, se toma un pedido de un lote y se trata de insertar en cada uno de los otros lotes disponibles, realizando la inserción únicamente si el pedido cabe en el lote. En concreto, se realizará el primer movimiento factible que suponga una mejora en valor de la función objetivo de la nueva solución vecina. El procedimiento se repite hasta que ningún movimiento mejora la solución.

ALGORITMO II
FUNCIÓN PRIMERAMEJORA

```
Función PrimeraMejora(x)
 1: repeat
 2:    x' ← x;
 3:    i←0;
 4:    repeat
 5:       i←i+1
 6:       x←argmin{ f (x), f (x^i)}, x^i  ∈ N(x)
 7:    until ( f (x) < f (x^i) or i = |N(x)|)
 8: until ( f (x) ≥ f (x'))
 9: return x
```

El método encargado de realizar una perturbación de la solución (denominado Shake) se utiliza para escapar del óptimo local actual mediante el movimiento a una nueva vecindad. En este caso, el método de perturbación se basa en movimientos de intercambio de pedidos. Es decir, se toman dos pedidos de dos lotes distintos, al azar, y se intercambian, siempre y cuando al realizar el intercambio la solución continúe siendo factible (no se sobrepase la capacidad de los lotes involucrados). Este mecanismo se aplica tantas veces como indique la variable *k*. Por último, se incluye un mecanismo para determinar si se debe explorar una vecindad de mayor tamaño, o bien se deben explorar de nuevo vecindades próximas a la que se acaba de explorar. Este mecanismo se implementa en el método CambioVecindad, recogido en el pseudocódigo de Algoritmo III.

ALGORITMO III
FUNCIÓN CAMBIOVECINDAD

```
Función CambioVecindad (x, x', k)
 1: if f (x') < f(x) then
 2:    x ← x' /* Realizar un movimiento */
 3:    k ← 1 /* Volver a vecindad inicial */
 4: else
 5:    k ← k+1 /* Siguiente vecindad */
```

Tal y como se ha indicado anteriormente, los pedidos no están disponibles, en su totalidad, al comienzo de la jornada de trabajo. Durante la ejecución del algoritmo los pedidos entran en el sistema siguiendo una distribución uniforme. El algoritmo BVNS se ejecuta durante un tiempo *t_max*, considerando, en cada iteración, los pedidos que hayan llegado hasta ese momento. Una vez alcanzado el valor de *t_max*, el método comienza de nuevo, realizando una nueva construcción y añadiendo los nuevos pedidos que hayan llegado durante el tiempo de la última ejecución de BVNS. En cada ejecución del algoritmo se devuelve la mejor solución encontrada. Cuando el trabajador encargado de la recogida ha terminado con un último lote, se le asigna, de la última solución disponible, uno de los lotes aún no recogidos. Para determinar qué lote será el siguiente en recogerse se emplea un algoritmo de selección. En concreto, este algoritmo, elegirá el lote que tiene un menor valor del cociente entre el tiempo estimado que tarda en recogerse el lote y el peso del mismo.

V. Resultados

En esta sección se describe el conjunto de instancias utilizadas para la validación de los algoritmos propuestos y se recogen, además, los resultados por instancia, de entre el conjunto de instancias seleccionadas.

A. *Instancias*

Las instancias usadas para hacer esta investigación son las publicadas por Albareda-Sambola et al. [18], en 2009. Las instancias constan de 4 tipos de almacenes rectangulares con el depósito colocado en la esquina inferior izquierda o en el centro inferior del almacén. La distribución de los ítems en los almacenes de las instancias puede ser de dos tipos: ABC y aleatoria. Las distribuciones de tipo ABC dividen los productos en tres categorías, situando los productos más demandados en la categoría A, y los menos demandados en las categorías B y C respectivamente. Una vez clasificados los productos, aquellos que caen dentro de la categoría A se sitúan más próximos al *depot,* los pertenecientes a la categoría B, se sitúan a continuación y, por último, se almacenan los productos pertenecientes a la categoría C. Además, cada instancia tiene distinto número de pedidos (50, 100, 150, 200 y 250). Las instancias empleadas para el problema son una pequeña selección de 10 instancias de entre todas las publicadas en [18]. Es importante destacar que, en este artículo no se emplean las instancias de tamaño 50 pedidos, ya que no generan la congestión suficiente para que sistema tenga sentido. Al no producirse congestión, para este tipo de instancias basta con seguir una política de recogida *First-Come-First-Serve* (FCFS). Para hacer la experimentación presentada en este documento, las instancias, con origen en la versión *offline* del problema, han tenido que ser adaptadas a instancias aptas para la recepción de pedidos *online*, generando un sistema de entrega de paquetes que distribuya la llegada de los pedidos. En concreto, el proceso encargado de entregar los paquetes al sistema sigue una distribución uniforme y los pedidos son entregados en un horizonte temporal de 4 horas.

B. *Resultados*

Por último, se ha comparado la propuesta algorítmica con los resultados obtenidos por el algoritmo del estado del arte "*Iterated Local Search"* propuesto por Henn [5] en su artículo de 2012. La comparación se ha realizado entre dicho algoritmo y dos variantes del BVNS propuesto, donde se combina un BVNS con cada uno de los dos tipos de construcciones iniciales. Un FCFS estándar y un FCFS-Completo, donde siempre se intenta completar la capacidad del lote a la llegada de cada pedido, con independencia de que dicho lote se hubiera dado por lleno en la iteración anterior. El BVNS se ejecuta en iteraciones de 60 segundos, permitiendo a los algoritmos constructivos, reconstruir la solución con los nuevos pedidos llegados al sistema antes de continuar con la búsqueda. Como selector, para determinar el siguiente lote a ser recogido, se emplea el criterio del lote que tiene un menor valor en el cociente obtenido al dividir el tiempo que tarda en recogerse el lote entre el peso del mismo.

En la Tabla I y en la Tabla II de resultados, respectivamente, se observa como la diferencia entre los resultados arrojados por los algoritmos es muy pequeña. En concreto, se reportan en la Tabla I los valores de función objetivo por instancia, obtenidos por cada método comparado y, en la Tabla II, la desviación obtenida respecto a la mejor solución del experimento. En este caso, ILS aparece como el mejor método, siendo capaz de obtener el mayor número de mejores soluciones y la menor desviación. Los mejores resultados, por cada instancia, están resaltados en negrita en las tablas.

TABLA I
Comparación Con El Estado Del Arte
Tiempo Máximo De Los Pedidos En El Sistema

Instancia	*ILS*	*BVNS - FCFS*	*BVNS – FCFS Completo*
A1_100_000	1721232	1666627	**1578027**
A1_100_030	**1215180**	2081531	2621530
A1_100_060	**1515669**	1945930	1885316
A1_100_090	**1099946**	1409978	1443535
A1_150_000	**2876411**	4215847	3375709
A1_150_030	**1081205**	1426639	1645529
A1_150_060	**2651687**	2855052	3770849
A1_150_090	**1232818**	1437467	1422996
A1_200_000	5578399	2273878	**1479272**

TABLA II
Comparación Con El Estado Del Arte
Desviación Respecto a la Mejor Solución Del Experimento

Instancia	*ILS*	*BVNS - FCFS*	*BVNS – FCFS Completo*
A1_100_000	9,1%	5,6%	**0,0%**
A1_100_030	**0,0%**	71,3%	115,7%
A1_100_060	**0,0%**	28,4%	24,4%
A1_100_090	**0,0%**	28,2%	31,2%
A1_150_000	**0,0%**	46,6%	17,4%
A1_150_030	**0,0%**	31,9%	52,2%
A1_150_060	**0,0%**	7,7%	42,2%
A1_150_090	**0,0%**	16,6%	15,4%
A1_200_000	277,1%	53,7%	**0,0%**

En la Tabla III, se muestran los resultados obtenidos de manera resumida. En concreto, se presenta el promedio de desviación de las 10 instancias consideradas y la suma del número de veces que un algoritmo ha encontrado el mejor valor.

TABLA III
Comparación Con El Estado Del Arte
Resultados Agrupados

Instancia	*ILS*	*BVNS - FCFS*	*BVNS – FCFS Completo*
Promedio	**31,8%**	32,2%	33,2%
Número de mejores	**7**	0	2

VI. Conclusiones

En este trabajo se ha estudiado el problema de agrupación y recogida de pedidos *online* en almacenes logísticos. Este problema de optimización, consiste, por un lado, en la agrupación en lotes de los pedidos que llegan al sistema y, por otro, en el diseño de rutas eficientes para recoger cada lote. La variante abordada considera que únicamente se dispone de un

trabajador y que los pedidos llegan al almacén de manera *online*, es decir, no todos los pedidos están disponibles al comienzo de la jornada de trabajo, sino que van llegando a medida que avanza el proceso de recogida.

Para abordar el problema se ha repasado el estado del arte de la variante *online* del mismo y se han descrito los métodos de *routing* más comunes en la literatura. A continuación, se ha propuesto un algoritmo basado en la metodología de Búsqueda de Vecindad Variable, en concreto, su versión Básica. Se ha implementado también el algoritmo más destacado del estado del arte. Para validar el algoritmo propuesto se ha adaptado un subconjunto de las instancias presentes en la literatura para la versión *offline* del problema, estableciendo una distribución uniforme para la llegada de los pedidos al almacén, y se han ejecutado ambos algoritmos en la misma máquina.

Pese a ser aún un trabajo en desarrollo, los resultados obtenidos son prometedores, siendo el algoritmo propuesto competitivo con la mejor variante en el estado del arte de la actualidad. Si bien, el método del estado del arte, propuesto por Henn en 2012, es el mejor, en promedio, de los algoritmos estudiados.

Como trabajos futuros, en primer lugar, se plantea la posibilidad de mejorar el algoritmo constructivo voraz utilizado como punto de partida inicial. Además, parece recomendable probar diferentes vecindades a la propuesta en este trabajo, de modo que se puedan conformar nuevas búsquedas locales para embeber dentro del esquema de Búsqueda de Vecindad Variable Básica. De manera similar, varias búsquedas locales pueden ser utilizadas en un esquema de Búsqueda de Vecindad Variable General. Por otro lado, las estrategias más prometedoras propuestas, pueden ser empleadas también para resolver otras variantes del problema tales como: disponer de más de un trabajador para recoger los pedidos en el almacén; o establecer una fecha/hora límite de entrega de cada pedido que se debe cumplir para evitar penalizaciones.

Agradecimientos

Esta investigación ha sido parcialmente financiada por el Ministerio de Economía y Competitividad, Referencias "TIN2015-65460-C2-2-P y MTM2015-63710-P".

Referencias

[1] R. Hall., «Distance approximation for routing manual pickers in a warehouse», *IIE Transactions,* nº 25, pp. 77–87, 1993.

[2] C. Petersen, «Routeing and storage policy interaction in order picking operations», *Decision Science institute Proceedings,* nº 31, pp. 1614 - 1616, 1995.

[3] R. De Koster, E. Van der Poort y K. Roodbergen, «When to apply optimal or heuristic routing of orderpickers, in: Advances in Distribution Logistics», *Springer* , pp. 375-401, 1998.

[4] K. Roodbergen y C. Petersen, «How to improve order picking efficiency with routing and storage policie», *Progress in Material Handling Practice,* pp. 107–124, 1999.

[5] S. Henn, «Algorithms for On-line Order Batching in an Order- Picking Warehouse», *Computers &OperationsResearch,* nº 39, pp. 2549–2563, 2012.

[6] H. R. Lourenco, O. C. Martin y T. Stutzle, «Iterated Local Search: framework and applications», *Handbook on MetaHeuristics,* pp. 363-397, 2001.

[7] C. G. y W. J., «Scheduling of vehicles from a central depot to a number of delivery points», *Operations Research,* vol. 4, nº 12, pp. 568-581, 1964.

[8] R. Pérez-Rodríguez, A. Hernández-Aguirre y S. Jöns, «A continuous estimation of distribution algorithm for the online order-batching problem», *Int J Adv Manuf Technol,* pp. 569–588, 2015.

[9] P. Larrañaga y J. A. Lozano, Estimation of Distribution Algorithms a New Tool for Evolutionary Computation, Boston, MA: Springer, 2002.

[10] F. Glover, «Future paths for integer programming and links to artificial intelligence», *Computers and Operations Research,* vol. 13, nº 5, pp. 533-549, 1986.

[11] J. Rubrico, T. Higashi, H. Tamura y J. Ota, «Online rescheduling of multiple picking agents for warehouse management», *Robotics and Computer Integrated Manufacturing,* nº 27, pp. 62–71, 2011.

[12] R. Graham, «Bounds for certain multi-processing anomalies», *Bell Syst Tech,* pp. 1563–1581, 1966.

[13] J. Aspnes, Y. Azar, F. A., S. Plotkin y O. Waarts, «On-line load balancing with applications to machine scheduling and virtual circuit routing», *Proceedings of the 25th A C Mannual symposium on the theory of computing,* pp. 623–631, 1993.

[14] J. Zhang, X. Wanga y K. Huang, «Integrated on-line scheduling of order batching and delivery under B2C e-commerce», *Computers & Industrial Engineering,* nº 94, pp. 280–289, 2016.

[15] Y. C. Ho y Y. Y. Tseng, «A study on order-batching methods of order-picking in a distribution centre with two cross-aisles», *International Journal of Production Research,* vol. 44, pp. 3391-3417, 2006.

[16] R. De Koster, E. S. Van der Poort y M. Wolters, «Efficient order batching methods in warehouses», *International Journal of Production Research,* vol. 37, pp. 1479–1504, 1999.

[17] N. Mladenovic y P. Hansen, «Variable neighborhood search», *Computers and Operations Research,* nº 24, pp. 1097–1100, 1997.

[18] M. Albareda-Sambola, A. Alonso-Ayuso y E. Molina, «Variable neighborhood search for order batching in a warehouse», *Asia-Pacific Journal of Operational Research,* vol. 26, nº5, pp. 655-683, 2009.