# Búsqueda de Vecindad Variable General Aplicada al Proceso de Recogida de Productos en Almacenes

Borja Menéndez, Eduardo G. Pardo, and Abraham Duarte

Dept. Informática y Estadística, Universidad Rey Juan Carlos,
Móstoles, Madrid, España
{borja.menendez,eduardo.pardo,abraham.duarte}@urjc.es
http://grafo.etsii.urjc.es/

**Resumen** El Problema del Empaquetado de Pedidos forma parte del proceso de recogida de productos en un almacén. Un conjunto de productos conforma un *pedido*, mientras que un conjunto de pedidos conforma un *lote*. El problema consiste en agrupar los pedidos recibidos en un almacén en diferentes lotes. Cada lote se recoge por un único trabajador sin exceder una capacidad límite determinada por el peso o volumen de los productos que un trabajador puede transportar a la vez. El objetivo consiste en minimizar el tiempo total necesario para recoger todos los productos. En este artículo se propone un algoritmo basado en la metodología de Búsqueda de Vecindad Variable General para abordar el problema. La aproximación propuesta mejora métodos previos del estado del arte.

**Keywords:** Búsqueda de Vecindad Variable General, GVNS, Problema del Empaquetado de Pedidos

## 1. Introducción

La industria ha encontrado un problema muy importante en el almacenamiento de productos en los últimos años, ya que es parte importante de la cadena de suministro. El almacenamiento se centra en controlar el movimiento y depósito de productos en un almacén, así como en procesar las transacciones asociadas como el envío, la recepción y la recogida de productos. Se han propuesto varias políticas en la literatura dirigidas a la mejora de las operaciones en almacenes. Algunas de ellas se pueden modelar como problemas de optimización [15]. En un nivel estratégico, el diseño del almacén es una de las decisiones más importantes [7]. De hecho, es un componente clave de las posteriores tareas y, entre otros, tiene un gran impacto en la longitud del recorrido de recogida en un almacén [11]. Desde un punto de vista táctico, la principal decisión es la política de almacenaje, es decir, decidir dónde ubicar cada producto. Existen políticas de almacenaje tradicionales como la aleatoria, la basada en demanda y la basada en clases, entre otras. Finalmente, las políticas operacionales están relacionadas

con la recogida de pedidos. Es decir, el proceso de recogida de productos de sus ubicaciones, en respuesta a una solicitud específica de un cliente. El lector puede acceder a una revisión más profunda en [10].

Este artículo se ocupa de las operaciones de recogida de pedidos, uno de los procesos más importantes en un almacén. El coste de este proceso puede ser más del 50 % del coste total de operaciones [2,5]. Un almacén recibe varios pedidos a diario de sus clientes. Cada pedido consiste en una lista de uno o más productos que se tienen que recoger del almacén y enviar a un cliente en concreto. Así, los productos deben ser recogidos por un trabajador. Se han propuesto varias estrategias en la literatura. Por ejemplo, la recogida de un único pedido en cada ruta; o la recogida de productos de manera indiscriminada y el posterior empaquetado y ordenación de los mismos. En particular, este artículo se centra en situaciones donde varios pedidos se agrupan en lotes, cumpliendo una restricción de capacidad fija determinada por el dispositivo de recogida. Entonces, cada lote se asigna a un trabajador, quien recoge en un único viaje todos los productos incluidos en los pedidos agrupados en el lote. Esta estrategia de recogida de pedidos se suele denominar *empaquetado de pedidos*.

De acuerdo con [4], es posible reducir el tiempo de recogida hasta un 35 % diseñando apropiadamente las rutas de los trabajadores. Por otra parte, si las dos decisiones relacionadas con la recogida de pedidos (es decir, empaquetado y enrutamiento) se consideran simultaneamente, los beneficios asociados pueden aumentarse considerablemente. Este hecho ofrece la motivación para el estudio del problema de optimización, conocido en la literatura como Problema de Empaquetado de Pedidos (OBP, del inglés *Order Batching Problem*), donde se combinan estas dos decisiones. Básicamente consiste en agrupar un conjunto de pedidos en lotes (con un límite de capacidad) y después, para cada lote, se define una ruta para recoger los pedidos correspondientes. Así, la función objetivo considera la configuración de los lotes que minimiza el tiempo necesario para recoger todos los productos. Se ha probado que el OBP es un problema $\mathcal{NP}$-difícil para instancias genéricas; no obstante, es resoluble en tiempo polinomial si cada lote no contiene más de dos pedidos [6]. Desafortunadamente, las instancias de almacenes reales no suelen caer en esta categoría. En consecuencia, este problema se ha aproximado de manera heurística en los últimos años.

La estrategia Primero en Llegar Primero en ser Servido (FCFS, del inglés *First-Come First-Served*) puede ser la primera aproximación heurística implementada en almacenes para asignar pedidos a lotes. En particular, se ordenan los pedidos de acuerdo a su hora de llegada al almacén. Siguiendo este criterio, los primeros pedidos que llegan son asignados al primer lote hasta que se exceda su capacidad. El pedido que sobrepase esta capacidad es asignado al siguiente lote, y así sucesivamente hasta que todos los pedidos son asignados a un lote. Una vez que un lote está conformado, puede pasar a ser recogido. Esta estrategia se ha usado extensamente debido a su simplicidad. En [3] se puede encontrar una clasificación de otras heurísticas sencillas de empaquetado. Entre otros, los autores resaltaron los "métodos semilla" [8,13,17] y los "métodos de ahorro" [19].

El primer algoritmo metaheurístico aplicado al problema fue descrito en [14]. Los autores propusieron un Algoritmo Genético para abordar el problema del empaquetado para almacenes con diferentes diseños.

Más tarde, en [1] se propuso un algoritmo basado en la metodología de Búsqueda de Vecindad Variable (VNS, del inglés *Variable Neighborhood Search*). Concretamente, propusieron seis vecindades diferentes en un esquema de Búsqueda de Vecindad Variable Descendente (VND, del inglés *Variable Neighborhood Descent*). El método constructivo asigna cada pedido a un lote diferente para, después, reducir el número de lotes juntándolos. Este proceso de juntar pedidos en lotes se repite hasta que no se puedan seguir agrupando. Una vez los lotes se conforman, se aplica un VND usando tres estrategias de enrutamiento conocidas (S-Shape [9], Largest Gap [3], y Combined [18]) para determinar la longitud del camino durante la recogida.

En este artículo se propone un algoritmo basado también en la metodología VNS para abordar el OBP. En concreto, se centra la atención en la estrategia de empaquetado, haciendo uso de un algoritmo de enrutamiento que evalúa las soluciones. El resto del artículo está organizado como sigue. En la Sección 2 se presenta el problema, mientras que en la Sección 3 se describe un algoritmo VNS para abordarlo. Finalmente, los experimentos y las conclusiones se exponen en las Secciones 4 y 5, respectivamente.

## 2. Problema del Empaquetado de Pedidos

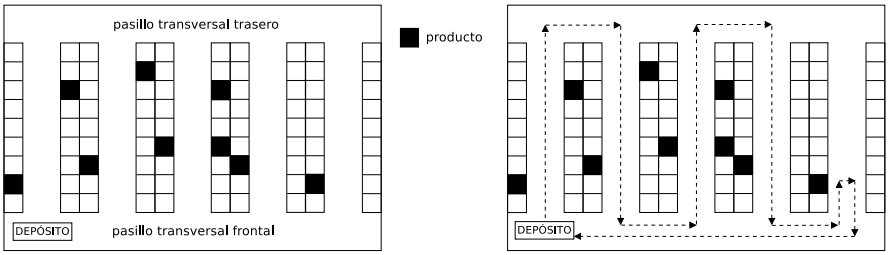
En el OBP, un producto representa un objeto que debe ser recogido de un almacén. Un pedido es un conjunto de productos agrupados que deben ser recogidos por el mismo trabajador. Dado un conjunto de pedidos recibidos en un almacén, existen dos estrategias de recogida de pedidos básicas: *por orden estricto* y *empaquetado*. En la primera, cada trabajador recoge todos los productos incluidos en un pedido, después se le asigna otro pedido a recoger, y así sucesivamente. En la segunda, varios pedidos se agrupan en lotes. A cada trabajador se le asigna un lote y este recoge todos los productos de los pedidos incluidos en él. Puesto que se recogen simultáneamente varios pedidos, se pueden diseñar rutas eficientes para reducir los tiempos de recogida de pedidos.

En este artículo se centra la atención en la estrategia de empaquetado de pedidos. En concreto, se empleará la estrategia de agrupar todos los pedidos en lotes, teniendo en cuenta una restricción de capacidad. Dicha capacidad está determinada por el dispositivo que utilice el trabajador para recoger los pedidos (denominado habitualmente «carrito»). El propósito del artículo es diseñar un algoritmo que conforme los lotes con el objetivo de minimizar el tiempo total necesario para recogerlos. Es importante hacer notar que no se aborda el problema de enrutamiento asociado al proceso de recogida, ya que se ha estudiado ampliamente en la literatura [4,18]. No obstante, es imprescindible disponer de un algoritmo de enrutamiento para poder evaluar la calidad de la solución propuesta. De esta forma se hará uso de un algoritmo de enrutamiento eficiente disponible previamente.

Existen diferentes variantes del OBP. En concreto, se considera la variante con solo un trabajador, en un almacén que tiene pasillos paralelos con la misma longitud. Dichos pasillos están conectados por pasillos transversales en la parte frontal y la parte trasera de cada pasillo paralelo. Con este diseño es posible realizar rutas por ambos lados de cada pasillo. De manera adicional, el depósito donde el trabajador debe dejar los productos recogidos es el punto de inicio y final de las rutas. Este depósito se puede localizar en cualquiera de las esquinas del almacén o en el centro de alguno de los pasillos transversales. La Figura 1a ilustra un almacén con 5 pasillos paralelos, 2 pasillos transversales y un depósito en la esquina inferior izquierda. Adicionalmente se representan con cuadrículas negras las posiciones que contienen productos que deben ser recogidos.

Aunque se ha probado que algunos esquemas de enrutamiento son óptimos, en la práctica conducen a rutas poco intuitivas para los trabajadores, pudiendo incluso generar congestiones en los pasillos. Es por eso que esquemas heurísticos más sencillos, como S-Shape [3] o Largest Gap [3], suelen predominar en la práctica.

La primera de ellas, S-Shape, es quizá la más sencilla de todas. El trabajador atraviesa por completo todos los pasillos donde hay algún producto que recoger. Si el número de pasillos fuese impar, el último pasillo solo se atraviesa hasta el producto más alejado, sin necesidad de recorrer el pasillo entero. En la Figura 1b se muestra un ejemplo de recogida de productos con esta estrategia en el almacén descrito en la Figura 1a. En ella se puede ver cómo el trabajador recorre enteros todos los pasillos en los que hay productos que recoger, salvo el último, que se recorre únicamente hasta el producto más alejado del pasillo frontal.



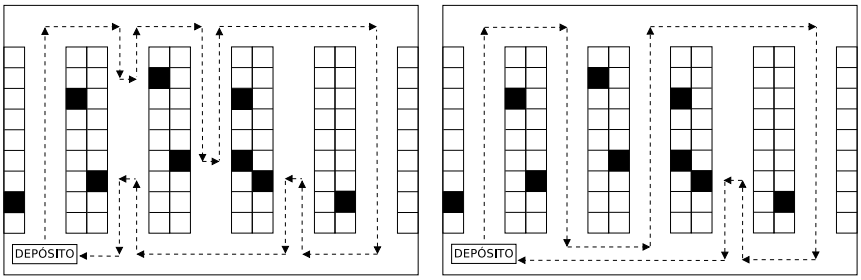

(a) Diseño del almacén. (b) Ruta obtenida con S-Shape.

Figura 1: Diseño del almacén y S-Shape para una configuración de productos.

En la estrategia de enrutamiento Largest Gap se recorren enteros tanto el primer como el último pasillo que contenga productos a recoger. El trabajador recorre el resto de pasillos sin atravesar el denominado «*largest gap*». Este *largest gap* se define como el mayor espacio entre cada par de productos consecutivos a recoger dentro de un mismo pasillo, o la distancia entre los pasillos transversales y los productos más cercanos a ellos. En la Figura 2a se muestra un ejemplo del

recorrido a seguir utilizando esta estrategia para el mismo almacén presentado en la Figura 1a. En ella se puede ver que el trabajador recorre tanto el primer como el último pasillo en los que hay productos que recoger. El resto de pasillos se recorren enteros, salvo la zona del mayor hueco. Por ejemplo, en el segundo pasillo, el trabajador entra y sale tanto por el pasillo frontal como por el trasero, puesto que el *largest gap* queda en el centro del pasillo. En el tercer pasillo, el trabajador entra y sale solamente por el pasillo trasero, mientras que en el cuarto pasillo solo lo hace por el pasillo frontal.

En concreto, en este artículo se usa una combinación de las dos estrategias previamente mencionadas (S-Shape y Largest Gap). Esta estrategia se denomina Combined y fue propuesta por Roodbergen y De Koster [18]. La estrategia Combined usa ideas de ambas: los pasillos se recorren enteros o bien se entra y se sale por el mismo pasillo transversal. El criterio para saber cuándo un pasillo se recorre entero o no es comprobar pares de pasillos adyacentes (donde el término "adyacente" se refiere únicamente a los pasillos que tienen al menos un producto que recoger). La opción a elegir se selecciona teniendo en cuenta cuál proporciona la ruta más corta, cuando se combina con la mejor opción para el siguiente pasillo. En la Figura 2b se representa la ruta ofrecida por esta estrategia para la misma configuracón de almacén expuesto en la Figura 1a. En este ejemplo, el método Combined guía al trabajador recorriendo el segundo y tercer pasillos enteros, realizando la estrategia S-Shape. En cambio, el cuarto pasillo se recorre solamente hasta el *largest gap*, realizando la estrategia Largest Gap.




(a) Ruta obtenida con Largest Gap. (b) Ruta obtenida con Combined.

Figura 2: Largest Gap y Combined para la misma configuración de productos.

## 3. Búsqueda de Vecindad Variable General

La Búsqueda de Vecindad Variable es una metaheurística basada en cambios sistemáticos de estructuras de vecindad [16]. Esta idea ayuda a escapar de óptimos locales y explorar otras zonas del espacio de soluciones. Existen diferen-

tes esquemas basados en esta metodología. Concretamente, se ha seleccionado General VNS (GVNS) para abordar el OBP.

El algoritmo propuesto se presenta en el Algoritmo 1. Este empieza con una solución factible, $s$, dada por un algoritmo constructivo (en este caso es una solución aleatoria). Existen dos parámetros más que configuran el algoritmo: $maxiter$ es el número máximo de iteraciones y $k_{max}$ es el número máximo de vecindades a explorar. En cada iteración, la solución se perturba de manera aleatoria con el procedimiento *Perturbacion* con tamaño $k$ (paso 5), obteniendo una nueva solución, $s'$, y se aplica un proceso de búsqueda local basado en VND, obteniendo una solución mejorada $s''$ (paso 6). Si $s''$ es mejor que $s$, $s$ se actualiza a $s''$ como la mejor solución encontrada hasta el momento y $k$ se pone a 1; en cualquier otro caso, $k$ se incrementa. Este proceso se repite hasta que se alcance $k_{max}$, siempre que no se haya llegado al máximo número de iteraciones permitidas. Por lo tanto, $maxiter$ y $k_{max}$ son parámetros de la búsqueda.

**Algoritmo 1** Algoritmo GVNS

```
1: procedure GVNS(s, maxiter, k_max)
2:     for i ← 0; i < maxiter; i ← i + 1 do
3:         k ← 1
4:         repeat
5:             s' ← Perturbacion(f, k)
6:             s'' ← VND(f')
7:             if s'' < s then
8:                 s ← s''
9:                 k ← 1
10:            else
11:                k ← k + 1
12:            end if
13:        until k > k_max
14:    end for
15:    return s
16: end procedure
```

No se reportan los pseudocódigos de los procedimientos *Perturbacion* y VND, ya que se ha seguido su implementación clásica [12]. En concreto, se han diseñado dos estructuras de vecindad diferentes para este problema. La primera de ellas se denomina $N_1$ y está basada en movimientos de inserción, $Insert$. En ella, un pedido $O_i$ se elimina de su lote actual, $B_j$, y se añade a otro ($B_k$) en el que el peso total de sus pedidos más el del pedido nuevo no exceda el límite de capacidad del carrito.

La segunda estructura de vecindad es la denominada $N_2$, basada en intercambios de pedidos (movimientos $Swap$). En este caso, dos pedidos ($O_i$ y $O_j$) de diferentes lotes ($B_k$ y $B_l$, respectivamente) se intercambian, de manera que el primer pedido ($O_i$) se elimina de su lote actual ($B_k$) para ser añadido al lote $B_l$ y, análogamente, $O_j$ se elimina de $B_l$ y se inserta en $B_k$. Destacar que este

movimiento tiene que ser factible también. Esto quiere decir que solo se puede realizar este movimiento si el peso de cada lote no supera la capacidad máxima del carrito tras el movimiento.

La fase de perturbación se introduce normalmente en VNS como una estrategia efectiva para escapar de un cuenco de atracción. Dada una solución $s$, el procedimiento de perturbación genera aleatoriamente una nueva solución $s'$ en la vecindad $k$-ésima (denotada como $N_k(s)$) aplicando $k$ movimientos. Concretamente, $N_k(s)$ contiene el conjunto de soluciones que pueden ser alcanzadas aplicando $k$ movimientos consecutivos.

Para este problema, el movimiento aplicado en el algoritmo GVNS para la estrategia de perturbación está basado en movimientos de intercambio ($Swap$). En concreto, el procedimiento de perturbación aplica un $Swap$ aleatorio $k$ veces consecutivas.

En este artículo se propone un método VND como el proceso de búsqueda local. Este VND explora las dos vecindades anteriormente descritas con una estrategia de primera mejora. En ambas vecindades, $N_1$ y $N_2$, solo se consideran los movimientos factibles que mejoran el valor de la función objetivo. Una vez que las dos vecindades previas se han definido, se propone una combinación de ambas en un método VND. El método recibe como parámetro de entrada una solución $s$. Las vecindades se ordenan de manera que se explora primero $N_1$ y después se explora $N_2$. Como es común en la comunidad VNS, las vecindades se exploran desde la más pequeña y rápida de explorar hasta la más grande. En este caso, $N_1$ (basada en movimientos de inserción) es normalmente más pequeña que $N_2$ (basada en movimientos de intercambio). Esto es debido a que el número de lotes es normalmente más pequeño que el número de pedidos, el orden dentro del lote no se tiene en cuenta y, además, solo se consideran los movimientos factibles. Es importante destacar que cuando se produce una mejora en una de las vecindades, el método explora nuevamente el espacio de soluciones partiendo de la primera vecindad. Este método termina cuando no hay mejoras en ninguna de las vecindades.

## 4. Experimentación

En esta sección se presentan los experimentos realizados para comparar empíricamente la propuesta con algoritmos previos del estado del arte. En concreto, se ha seleccionado el mejor algoritmo previo para el OBP [1]. Para realizar una comparativa justa entre los algoritmos se ha usado un subconjunto de 600 instancias propuestas en su artículo. Para más información se remite al lector a la descripción del Almacén 1 en la Sección 5.1 de dicho artículo.

Del conjunto total de instancias se han seleccionado 20 de ellas representativas para ajustar los parámetros del GVNS. De acuerdo con los resultados obtenidos, las variables $k_{max}$ y $maxiter$ se han establecido a 5, ya que valores mayores incrementarían el tiempo de ejecución y la mejora obtenida en los resultados resulta marginal.

En la Tabla 1 se reportan los resultados obtenidos por el GVNS y por el método propuesto en [1], cuyo nombre es "Exchange". En ella se presentan la media del valor de la función objetivo (*Media*), la media de la desviación respecto al mejor valor conocido (*Desv. (%)*), la media del tiempo de ejecución medida en ms (*CPUt (ms)*) y el número de mejores soluciones encontradas (*#Mej.*). Ambos métodos se han implementado en Java 6 y han sido ejecutados en la misma máquina (Intel Core i7 con 3.4 GHz y 4 GB de RAM con Linux Mint Debian Edition de 64 bits). Además, ambos algoritmos usan la misma estrategia de enrutamiento anteriormente mencionada (Combined).

| | | **Media** | **Desv. (%)** | **CPUt (ms)** | **#Mej.** |
|---|---|---|---|---|---|
| 50 pedidos | GVNS | 3510.21 | 1.50 | 33.90 | 43 |
| | Exchange | 3501.24 | 1.42 | 11.42 | 45 |
| 100 pedidos | GVNS | 6625.09 | 0.64 | 96.83 | 52 |
| | Exchange | 6656.01 | 1.11 | 79.92 | 31 |
| 150 pedidos | GVNS | 9849.93 | 0.33 | 205.98 | 68 |
| | Exchange | 9918.03 | 1.06 | 250.17 | 22 |
| 200 pedidos | GVNS | 12934.16 | 0.18 | 358.82 | 84 |
| | Exchange | 13041.03 | 1.02 | 584.25 | 18 |
| 250 pedidos | GVNS | 16140.21 | 0.08 | 547.21 | 88 |
| | Exchange | 16268.48 | 0.95 | 1179.83 | 18 |
| Global | GVNS | 9811.92 | 0.55 | 248.55 | 335 |
| | Exchange | 9876.96 | 1.12 | 421.12 | 134 |

Tabla 1: Comparación de los algoritmos GVNS y Exchange.

Los resultados en la Tabla 1 están divididos en grupos teniendo en cuenta el número de pedidos de cada instancia. En concreto, se ha dividido el experimento en 5 grupos de 50, 100, 150, 200 y 250 pedidos con 120 instancias por grupo. Finalmente, también se muestran los resultados sobre el conjunto total de instancias (véase la fila *Global*). De acuerdo con los resultados de la Tabla 1, GVNS mejora el estado del arte en la mayoría de los subconjuntos en términos de *Desv. (%)* y *#Mej.*, así como en tiempo de ejecución (*CPUt (ms)*) y media del valor de la función objetivo (*Media*). La única excepción es el conjunto formado por instancias de 50 pedidos, donde el método Exchange obtiene resultados ligeramente mejores. Si se considera el conjunto total, el GVNS obtiene una desviación media de 0.55 % respecto de los mejores valores conocidos, mientras que el Exchange obtiene una desviación de 1.12 % en casi el doble de tiempo de ejecución que GVNS. También es destacable el número de mejores soluciones obtenidas por cada algoritmo, puesto que GVNS es capaz de obtener 335 mejores valores, mientras que Exchange obtiene 134.

Para terminar con los experimentos, se han llevado a cabo tests estadísticos para confirmar que las diferencias obtenidas por el algoritmo propuesto son esta-

dísticamente significativas. En concreto, se ha realizado un test de Wilcoxon para cada grupo de instancias de manera separada y también para todas las instancias juntas. El $p - valor$ obtenido por todos los subconjuntos es menor que 0.001, lo que indica que existen diferencias significativas entre los algoritmos evaluados, excepto para el subconjunto en el que hay 50 pedidos por cada instancia, en el que el $p-valor$ obtenido es de 0.743 (curiosamente es el único subconjunto donde el método Exchange mejora ligeramente a GVNS). Finalmente, si se consideran todas las instancias juntas, el $p - valor$ es, de nuevo, menor a 0.001, indicando que las diferencias entre ambos algoritmos son estadísticamente significativas.

## 5. Conclusiones

En este artículo se ha abordado el Problema del Empaquetado de Pedidos en un almacén con pasillos paralelos y dos pasillos transversales, uno en la parte frontal y otro en la parte trasera del almacén. En concreto, se ha propuesto un algoritmo GVNS basado en la utilización de dos vecindades diferentes para abordar el problema. El algoritmo propuesto se ha comparado experimentalmente con el mejor algoritmo previo del estado del arte sobre un conjunto de instancias de referencia. Los resultados indican que la propuesta mejora al método previo y que las diferencias son estadísticamente significativas.

## 6. Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad de España, a través del proyecto con referencia TIN2012-35632-C02-02, y la Comunidad de Madrid, a través del proyecto con referencia S2013/ICE-2894.

## Referencias

1. Albareda-Sambola, M., Alonso-Ayuso, A., Molina, E., Simón de Blas, C.: Variable neighborhood search for order batching in a warehouse. Asia-Pacific Journal of Operational Research. 1, 655–683 (2009)
2. Coyle, J.J., Bardi, E.J., Langley, C.J.: The management of business logistics. West Publishing Company Minneapolis/St. Paul. 6 (1996)
3. De Koster, R., Van der Poort, E., Wolters, M.: Efficient orderbatching methods in warehouses. International Journal of Production Research. 37 (7), 1479–1504 (1999)
4. De Koster, R., Roodbergen, K.J., Van Voorden, R.: Reduction of walking time in the distribution center of De Bijenkorf. Technical report, Rotterdam School of Management, Erasmus University of Rotterdam (1998)
5. Drury, J.: Towards more efficient order picking. IMM Monograph. 1 (1998)
6. Gademann, N., Van de Velde, S.: Order Batching to Minimize total Travel Time in a Parallel-Aisle Warehouse. IIE Transactions. 37 (1), 63–75 (2005)
7. Ghiani, G., Laporte, G., Musmanno, R.: Introduction to logistics systems planning and control. John Wiley & Sons (2004)

8. Gibson, D.R., Sharp, G.P.: Order batching procedures. European Journal of Operational Research. 58 (1), 57–67 (1992)
9. Goetschalckx, M., Ratliff, H.D.: Order picking in an aisle. IIE Transactions. 20 (1), 53–62 (1988)
10. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse operation: A comprehensive review. European Journal of Operational Research. 177 (1), 1–21 (2007)
11. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse design and performance evaluation: A comprehensive review. European Journal of Operational Research. 203 (3), 539–549 (2010)
12. Hansen, P., Mladenovic, N.: Variable neighborhood search: Principles and applications. European Journal of Operational Research. 130 (3), 449–467 (2001)
13. Ho, Y.-C., Tseng, Y.-Y.: A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. International Journal of Production Research. 44 (7), 3391–3417 (2006)
14. Hsu, C.-M., Chen, K.-Y., Chen, M.-C.: Batching orders in warehouses by minimizing travel distance with genetic algorithms. Computers in Industry. 56 (2), 169–178 (2005)
15. Karasek, J.: An overview of warehouse optimization. Computers in Industry. 2 (3), 111–117 (2013)
16. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research. 24 (11), 1097–1100 (1997)
17. Pan, C.-H., Liu, S.-Y.: A comparative study of order batching algorithms. Omega. 23 (6), 691–700 (1995)
18. Roodbergen, K.J., De Koster, R.: Routing methods for warehouses with multiple cross aisles. Internation Journal of Production Research. 39 (9), 1865–1883 (2001)
19. Rosenwein, M.B.: A comparison of heuristics for the problem of batching orders for warehouse selection. International Journal of Production Research. 34 (3), 657–664 (1996)