Discrete Applied Mathematics 161 (2013) 2862–2875

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

The deterministic product location problem under a pick-by-order policy

Nils Boysen *, Konrad Stephan

Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß -Straß e 3, 07743 Jena, Germany

ARTICLE INFO

Article history:
Received 12 April 2012
Received in revised form 26 June 2013
Accepted 2 July 2013
Available online 27 July 2013

Keywords:
Logistics
Warehousing
Product location
Order picking

ABSTRACT

This paper treats the product location problem in warehouses, i.e., stock keeping units (SKUs) are to be assigned to storage positions in order to minimize the resulting picking effort when retrieving SKUs in a pick-by-order environment. We restrict our view on warehouses having a single cross aisle and show that already very simple layouts consisting of only a single rack lead to NP-hard optimization problems. In addition to a complexity analysis for different layouts, elementary solution procedures are introduced and tested. Finally, we investigate the robustness of our deterministic problem when facing erroneous input data.

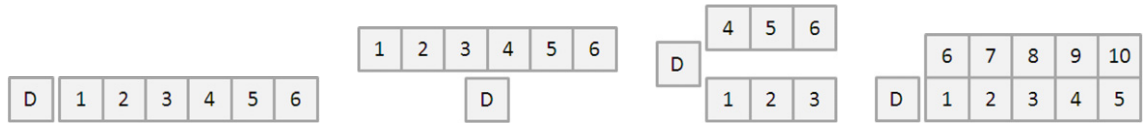
1. Introduction

Order picking is an elementary warehousing function dealing with the retrieval of stock keeping units (SKUs) from their storage locations in order to satisfy a given demand defined by picking orders. Typically, order picking is a labor-intensive task, which is both time-critical and cost-intensive. Therefore, it is not astounding that optimizing picking operations by computerized solution procedures has received (and is still receiving) plenty of attention among practitioners and academic researchers alike (e.g., see the latest survey papers on warehouse operations by de Koster et al. [5]; Gu et al. [10]). In this context, we consider the deterministic product location problem (also known as the slotting problem), which is defined as follows:

Consider a given set $P = \{1, \ldots, n\}$ of SKUs to be assigned to a given set of storage locations $S = \{s_1, \ldots, s_n\}$ arranged in racks of a given layout, so that each product receives exactly one storage location. Once SKUs are assigned they need to be picked according to a given set O containing m picking orders, where each order $o \in O$ consists of a subset of SKUs to be retrieved (e.g., all items ordered by a specific customer). Order picking is organized according to the pick-by-order policy (also denoted as discrete order picking), so that all SKUs of an order are jointly picked during a single tour of a picker. Capacity constraints are considered to be a non-issue, so that exactly m tours occur. A tour starts and ends in the depot (or input–output point) and consists of an ordered set of locations where the products of the respective order are stored. We restrict our view on warehouse layouts having only a single front-end cross-aisle, i.e., for reaching another storage aisle, so that pickers enter and leave storage aisles over the same entrance point at the front (also denoted as the "out and back" approach or the "return policy"). Note that our (front-end) cross-aisle should not be confound with a middle aisle crossing the storage area in order to introduce additional flexibility for the pickers changing storage aisles. We abstain from the latter type of (middle) cross-aisle, so that we restrict ourselves to what is also denoted as the one-block layout. Both assumptions, the one-block layout plus the "out and back" policy, make the integrated picker routing problem trivial to solve, i.e., only the

* Corresponding author. Tel.: +49 3641 943100.
E-mail addresses: nils.boysen@uni-jena.de (N. Boysen), konrad.stephan@uni-jena.de (K. Stephan).
URL: http://www.wiwi.uni-jena.de/pil/ (N. Boysen).

0166-218X/$ – see front matter
http://dx.doi.org/10.1016/j.dam.2013.07.002



(a) Single rack with front end depot. (b) Single rack with central depot. (c) Single aisle with front end depot. (d) Two-dimensional rack with front end depot.

Fig. 1. Investigated layouts for the storage location problem (PLP) ((a)–(c) top view and (d) side view).

farthest picking location per order and aisle has to be determined. Within this setting, our product location problem (PLP) aims to assign products to storage locations, so that the total walking distance over all given picking orders is minimized.

This paper investigates the computational complexity of the PLP in different warehouse layouts and presents suited solution procedures. Specifically, we investigate the most basic PLP-layout (depicted in Fig. 1(a)) consisting of a single straight rack and a front end depot. For this building block we prove NP-hardness in the strong sense and present exact and heuristic optimization procedures. Then, we show how these basic results can be applied for proving computational complexity for other layouts, i.e., layouts (b)–(d) in Fig. 1, and how the solution procedures can successfully be applied for repeatedly solving subproblems in real-world order picking settings.

The remainder of the paper is structured as follows. Section 2 provides a literature review on related research. Then, Section 3 investigates layout (a) where a single rack faces a front end depot. For this basic PLP, a complexity analysis is provided and exact and heuristic solution procedures are introduced, which are tested in a comprehensive computational study. The complexity results are then applied for proving NP-hardness of more complex layouts, i.e., layouts (b)–(d), within Section 4. Furthermore, we investigate a typical real-world setting for order picking and show our basic PLP being an important subproblem (Section 5). Section 6 investigates the robustness of PLP solutions if incomplete or erroneous order information is available and, finally, Section 7 concludes the paper.

2. Literature review

Numerous studies on optimizing warehouse operations in general and on product location and picker routing in particular exist. Instead of trying to summarize this vast body of literature we refer to the excellent survey papers published on these topics over the past years, e.g., [5,9,19,10]. To emphasize the importance of simultaneously planning product locations and picker routes, we only make one exception. In a recent case study at a distribution center for alcoholic beverages in Québec (Canada), Renaud and Ruiz [18] calculate an impressive reduction rate of 27% on the total picking effort when unifying both problems. With regard to the information availability, i.e., the question whether reliable information on (short term) picking orders exist when solving the (long term) product location problem, three cases can be distinguished:

- Whenever the composition of orders considerably varies over time, e.g., induced by highly volatile customer demands, orders are hardly predictable and *no reliable order information* is available. In this case, only product specific information is on hand, so that, for instance, the famous cube-per-order index (see [12]) can be applied for assigning storage positions to SKUs.
- In a more stable environment, reliable *product correlations* defining the probability of a product pair being jointly ordered may be obtained from historical data (see [7,3]). These correlations allow for a grouping of product families, so that similar SKUs are located in the same region of the storage area and travel distance during operational order picking can be reduced.
- This paper assumes a given *deterministic order set*. Clearly, this assumption seems unrealistic for volatile environments faced by most distribution centers serving demands of final customers. However, a given order set may be a valid approximation of reality in intermediate warehouses, where, for instance, recurrent orders for parts consumed by cyclically produced production lots in a low-variety make-to-stock supply chain are picked. Even in a truly stochastic environment the deterministic problem can contribute to the overall solution, e.g., in a scenario-based approach. Furthermore, solving the PLP with a deterministic order set can deliver benchmark solutions for the former two cases, in order to quantify the value of information from a theoretical point of view.

To the best of the authors' knowledge there exists only the study of Van Oudheusden and Zhu [20], who also treat a product location problem for a given set of recurrent orders. They briefly mention the basic PLP resulting from layout (a) in Fig. 1 as an interesting subproblem and present a very basic myopic procedure. A complexity analysis and exact solution procedures for our PLP are not provided. Instead, they focus a more complex layout with a two-dimensional rack and a front end depot (layout (d) in Fig. 1) applied in a man-aboard automated storage and retrieval system. They presuppose given routes for each order predetermined by some traveling salesman procedure, so that the routing problem is excluded, and provide a heuristic solution procedure for the product location problem. Up to now, the complexity status of this PLP has been unknown. In Section 4.3 we finally prove the PLP for layout (d) being strongly NP-hard.

With regard to its mathematical structure our basic PLP bears some similarities to the single-row equidistant facility layout problem (e.g., [14,15]), where facilities (e.g., are to be assigned to locations (equally spaced along a straight line)), so that the sum of product flows between facilities weighted with the distances of their assigned locations is minimized.

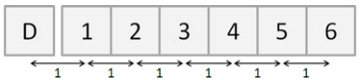


Fig. 2. Single rack with a front end depot.

One potential application for this problem is, for instance, the location of machines along some straight material handling device in a flexible manufacturing system (see [14]). However, due to diverging evaluation schemes of a given assignment, the results are not transferable between both problems.

3. Single rack with front end depot

3.1. Problem description and complexity analysis

In this section we treat the PLP in its very basic form and assume that all storage locations are arranged in a single rack along a straight line with the depot being located at the front end of the rack (see Fig. 2). Without loss of generality, we assume storage locations being numbered from 1 to n according to increasing distance from the depot (denoted as D within Fig. 2), which is located at position zero. Furthermore, we presuppose equidistant storage positions, so that we only need to count the number of storage positions passed during a picking tour. In this basic problem setting the routing subproblem is trivial, because the tour length per order is directly determined by the way to and from the one SKU assigned to the storage location farthest from the depot. Thus, the overall walking distance for a given assignment of products to storage locations amounts to: $Z = \sum_{o \in O} 2 \cdot \max_{i \in o}\{s(i)\}$, where $s(i)$ denotes the storage location SKU i is assigned to.

Layouts with a single rack are often chosen in forward areas, where fast-moving items are stored in easy-to-access racks, e.g., gravity flow racks, concentrated in a compact area (e.g., see [2]). Alternatively, this setting may occur as a subproblem in a more complex warehouse, e.g., consisting of multiple racks arranged at right angles along a picking belt (see Section 5).

Theorem 1. *The PLP with a single rack and a front end depot is NP-hard in the strong sense.*

The following transformation is from the linear arrangement problem (LAP), which is well-known to be NP-hard in the strong sense (see [8]).

LAP: Given a graph $G = (V, E)$ and a positive integer K. Is there a one-to-one-function $f : V \to \{1, 2, \ldots, |V|\}$, i.e., a numbering of nodes V with integer values from 1 to $|V|$, such that

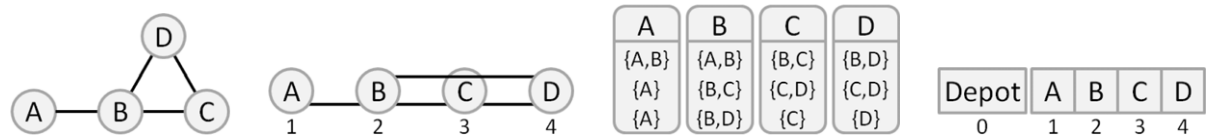
$$\sum_{(u,v) \in E} |f(u) - f(v)| \leq K? \tag{1}$$

Proof. Within our transformation of LAP to PLP we introduce a product and a storage location for each node, so that $n = |V|$. The integer value $f(u)$ assigned to a node u within LAP corresponds to the storage location $s(i)$ assigned to SKU i within PLP. Given the maximum degree $\delta(G)$ of the LAP graph, we introduce $\delta(G)$ orders for each node $u \in V$: First, an order $\{u, v\}$ is generated for each adjacent node v, so that for each edge $(u, v) \in E$ two orders $\{u, v\}$ and $\{v, u\}$ are generated. Then, for each node having a degree less than $\delta(G)$ we extend the order set by additional single-item-orders $\{u\}$ until $\delta(G)$ orders per node are generated. In total, $\delta(G) \cdot |V|$ single- and two-item-orders are introduced. The question we ask is whether we can find a solution for PLP with objective value $Z \leq 2K + \delta(G) \cdot |V| \cdot (|V| + 1)$.

The $\delta(G)$ orders associated with each SKU u are either single-item-orders, which require a walking distance of $2s(u)$, or two-item-orders. Each order $\{u, v\}$ of the latter kind always exists twice, because in the name of each edge (u, v) two identical orders are introduced, i.e., one when generating the $\delta(G)$ orders for node u and the other when generating orders for v. The travel required for any of these order pairs is twice the distance to and from the SKU stored farther from the depot. If $s(u) < s(v)$ this amounts to a distance of $4s(v)$. This distance can be split into the way $2s(v)$ to and from the back SKU plus the way $2s(u)$ to and from the front item plus the distance $2(s(v) - s(u))$ between both positions. If we assign the former two distances $2s(v)$ and $2s(u)$ to items v and u, respectively, then it becomes obvious that each storage position $i = 1, \ldots, n$ is to be visited exactly $\delta(G)$ times. Thus, we have an inevitable distance, i.e., independent of the assignment of SKUs to storage locations, of $2\delta(G) \cdot \frac{|V| \cdot (|V|+1)}{2}$. The remaining distance $2(s(v) - s(u))$ within PLP, which is dependent of the chosen storage locations for SKUs, exactly equals the difference in the node numbers assigned to each edge within LAP multiplied with constant 2, so that both problems are directly transferable from each other and the theorem holds. □

Example. For a more intuitive understanding consider the LAP instance given in Fig. 3(a) and its belonging optimal linear ordering (b) with an objective value of $K = 5$. The orders to be generated in PLP are depicted in (c). The inevitable walking distance amounts to $\delta(G) \cdot |V| \cdot (|V| + 1) = 3 \cdot 4 \cdot 5 = 60$ and in the corresponding optimal solution of PLP (d) the total objective value amounts to $Z = 70$.

Note that the special case of our basic PLP with $|o| = 1 \, \forall o \in O$ is solvable to optimality in $O(n \log n)$ by successively assigning storage positions to SKUs ordered in descending order according to their total demand (see Algorithm 1 in Section 3.3).



(a) LAP graph. (b) LAP solution. (c) Order set of SLP. (d) SLP solution.

Fig. 3. Example for the transformation of LAP to PLP.

3.2. An exact dynamic programming procedure

For determining optimal solutions we introduce a dynamic programming (DP) procedure, which resembles the traditional DP scheme for sequencing problems as defined by Held and Karp [11]. The decision process of the DP is subdivided into $n+1$ stages, where each stage $i = 1, \ldots, n$ represents a storage location (plus a virtual start stage 0). Any stage i contains a set of states, where each state represents a possible subset $Q \subseteq P$, $|Q| = i$, of SKUs assigned up to storage position i. The optimum schedule for subset Q is found according to the recursive equations

$$h^*(Q) = \min_{p \in Q} \left\{ h^*(Q \setminus \{p\}) + f\,(p, Q \setminus \{p\}) \right\} \tag{2}$$

with $h^*(\emptyset) = 0$. Here, the objective value of the optimal storage locations for products $Q \setminus \{p\}$ is added to the contribution $f(p, Q \setminus \{p\})$ of assigning SKU p to storage location i and having product set $Q \setminus \{p\}$ assigned to previous storage locations $1, \ldots, i-1$, where $f(\widetilde{p}, \widetilde{Q})$ is calculated according to

$$f(\widetilde{p}, \widetilde{Q}) = 2 \cdot (|\widetilde{Q}| + 1) \cdot |\{o \in O : o \setminus \widetilde{Q} = \{\widetilde{p}\}\}|.$$

The overall objective is to find optimal solution value $h^*(P)$ for the set of all SKUs being assigned. We can determine $h^*(P)$ by a stage-wise forward recursion using (2). Finally, when the final stage is reached and optimal solution value $h^*(P)$ is determined, a simple backward recursion can be applied to determine the optimal assignment.

There are 2^n states and $n \cdot 2^{n-1}$ transitions to be evaluated. Each transition requires the inspection of all m orders, which in the worst case contain all n products. Thus, the computational complexity of DP amounts to $O(m \cdot n^2 \cdot 2^n)$, so that (in line with the previous complexity result) there is an exponential increase of runtime in the number of storage locations n.

Example (*Cont.*)**.** The DP graph and four bold faced shortest paths with length $Z = 70$ representing all optimal assignments are depicted in Fig. 4. For instance, the lowermost path via D, CD and BCD represents the assignment $s(A) = 4$, $s(B) = 3$, $s(C) = 2$, and $s(D) = 1$.

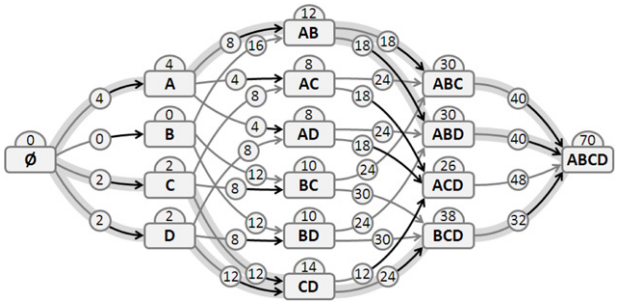


Fig. 4. DP graph for our example.

3.3. Extensions of DP and heuristic solution procedures

First, we show how to extend the DP approach presented in the previous section with upper and lower bounds to a bounded dynamic programming (BDP) approach. Then, we introduce two straightforward greedy heuristics.

DP can be extended to a BDP approach by additionally computing state specific lower bounds $LB(Q)$ on the remaining path length to the target state. Furthermore, a global upper bound UB can be determined upfront by some heuristic procedure. Specifically, within our BDP heuristic G2 described in subsequent paragraphs is applied. Whenever $LB(Q)$ plus the shortest path to the current state Q is equal to or exceeds upper bound UB, the respective state can be fathomed as it cannot be part of a solution with a better objective value than the incumbent solution. In the following it is shown how to determine a simple lower bound argument for any state.

This bound is based on the relaxation of the consistent assignment of SKUs among all orders, so that for each order separately the number u of yet unassigned products is assumed to be assigned to the next u storage locations and lower

bound $LB(Q)$ for a state defined by set Q of products assigned to the first $|Q| = i$ stages amounts to:

$$LB(Q) = 2i \cdot |\{o \in O : o \setminus Q \neq \emptyset\}| + 2 \cdot \sum_{o \in O} |o \setminus Q|. \tag{3}$$

Bound computation requires an effort of $O(nm)$ per state and preliminary tests revealed that LB is not tight enough to fathom states of earlier stages. Thus, bound computation is only applied at stages larger than $\frac{n}{2}$.

Example (*Cont.*)**.** For our example of Fig. 3, consider a state with $Q = \{A, B\}$ assigned to the first two storage locations and $h^*(Q) = 12$. The lower bound for this state amounts to $LB(Q) = 52$, so that the state could be fathomed if $UB \leq 64$, which, however, is not possible since the optimal solution value amounts to $Z = 70$.

The first simple greedy heuristic resembles the approach often applied in real-world applications, which is to relax the joint occurrences of products in orders and to successively assign SKUs to storage locations by sorting SKUs according to their total demand. This policy denoted as G1 is formalized by Algorithm 1.

Algorithm 1: Greedy heuristic G1

```
1 Initialization: Order the set of SKUs by their total demand in descending order resulting in vector π = π_1, ..., π_n;
2 for s = 1 to n do
3     assign SKU π_s to storage location s;
4 return assignment ;
```

Due to the sorting operation the runtime complexity of G1 is $O(n \log n)$. Calculating the associated objective value requires the identification of the each order's maximum storage position and, thus, requires $O(mn)$.

Example (*Cont.*)**.** Our example instance introduced in Fig. 3 results in the assignment of $s(A) = 4$, $s(B) = 1$, $s(C) = 2$, and $s(D) = 3$ and an objective value of $Z = 74$.

The second greedy heuristic denoted as G2 assigns product locations from a backward direction according to decreasing occurrences in the set of active orders, where the set of active orders is continuously updated by deleting those orders where the most recently added item has already been located. The procedure is formalized by Algorithm 2.

Algorithm 2: Greedy heuristic G2

```
1 Initialize the current set of active orders Q = O;
2 for s = 1 to n do
3     count the number of occurrences of each SKU in Q and assign SKU p' not yet assigned and having least occurrences to position s' = n − s + 1;
4     reduce the set of active orders by all those orders in which p' occurs;
5 return assignment ;
```

The runtime complexity of G2 is $O(mn^2)$.

Example (*Cont.*)**.** For our example the resulting assignment is $s(A) = 4$, $s(B) = 3$, $s(C) = 2$, and $s(D) = 1$, so that the optimal objective value of $Z = 70$ is reached.

3.4. Computational study

For evaluating the performance of our solution procedures, we first elaborate on instance generation. Our instance generator receives the following initial input data: the number n of SKUs (and storage location), and the number m of orders. All values assigned to both parameters are combined in a full-factorial design and per combination of parameters instance generation is repeated R times. Each single instance for a given parameter setting is generated by, first, randomly drawing (with equal distribution) the number $|o|$ of SKUs contained in an order o from the interval $[1; n]$ and, then, randomly drawing $|o|$ unique products. Note that the solution procedures described above have been implemented in C#.NET and run on a 2.1 GHz PC, with 2 GB of memory.

First, we investigate the solution performance for instances where BDP is able to determine optimal solution values. Therefore, our instance generator is initialized with parameter values $n \in \{4, 8, 12, 16\}$, $m \in \{10, 20, 30, 40\}$, and $R = 50$, so that in total 800 data instances are generated. Table 1 lists the solution time (CPU time in seconds) of BDP as well as the number of optimal solutions (#opt) gained, and the average and maximum relative gap (avg gap and max gap in percent) from the optimum for the heuristic procedures G1 and G2 in dependency of the parameters n and m. As a further benchmark we implemented a straightforward simulated annealing approach. The solution vector directly stores the assignment of SKUs to locations and we apply swap moves, i.e., two randomly drawn storage positions exchange their products, to generate neighboring solutions. A detailed description of this procedure denoted as SA^D is given in the Appendix.

With regard to the solution time the results confirm the exponential (linear) increase of BDP's runtime in n (m). For G1, G2, and SA^D solution time is barely measurable for these small instances. Even with $n = 1000$ SKUs and $m = 4000$ orders the

Table 1
Computational performance for a single rack and a front end depot (small instances).

n	m	BDP	G1			G2			SA^D		
		CPU time	#opt	avg gap	max gap	#opt	avg gap	max gap	#opt	avg gap	max gap
4	10	<0.01	34	1.6	12.9	49	0.1	3.6	50	0.0	0.0
	20	<0.01	20	2.0	10.3	46	0.2	3.4	50	0.0	0.0
	30	<0.01	26	1.0	7.8	46	0.2	3.3	50	0.0	0.0
	40	<0.01	20	1.3	5.8	42	0.3	3.3	50	0.0	0.0
8	10	0.01	1	7.6	17.7	28	1.4	8.3	50	0.0	0.0
	20	0.01	2	4.9	16.7	32	0.7	5.8	50	0.0	0.0
	30	0.02	1	3.6	8.8	32	0.6	5.0	50	0.0	0.0
	40	0.02	2	2.9	8.6	27	0.5	4.5	49	0.0	0.4
12	10	0.36	0	11.2	33.8	13	3.1	15.4	33	0.4	2.5
	20	0.54	0	6.9	12.4	11	1.3	5.2	17	0.6	2.0
	30	0.68	0	5.5	10.5	8	1.2	3.8	7	0.6	1.7
	40	0.80	0	5.5	10.4	15	0.8	4.0	1	0.6	1.3
16	10	74	0	14.1	25.7	10	3.5	19.0	2	1.9	4.8
	20	112	0	10.1	15.9	6	2.6	7.5	0	2.2	3.7
	30	132	0	7.2	14.0	1	1.7	5.5	0	2.1	3.6
	40	133	0	6.1	10.7	3	1.3	4.2	0	1.8	3.3

solution times of G1 and G2 do not exceed a single CPU second. With regard to the solution quality G2 outperforms G1. Over all instances, G2 produces an average relative gap from the optimum of only 1.2%, whereas G1 shows an average gap of 5.7%. However, this finding holds not true for any single instance. In eight instances G1 finds an even better solution than G2. Both procedures profit from an increasing number m of orders, because for $m = 10$ (except for $n = 4$) the gap of both heuristics is the highest. Among all heuristics SA^D performs best for the small instances. It produces an average gap of only 0.6%.

Additionally, we test our heuristics with larger instances, which cannot be solved to optimality by BDP. The data generator is initialized with parameter values $n \in \{20, 30, 40, 50\}$, $m \in \{200, 300, 400, 500\}$, and $R = 50$, so that an additional 800 large instances are generated. In neither instance G1 and G2 require more than 0.02 CPU seconds, whereas SA^D requires up to 15 CPU seconds ($n = 50$ and $m = 500$) and 7 CPU seconds on average. With regard to the solution quality, now, G2 performs best. In 792 of 800 (99%) instances G2 delivers the best solution value and in relation to the best solution found per instance the average gap over all instances for G1, G2, and SA^D amounts to 2.7%, 0.002%, and 1.7%, respectively. Obviously, the simple meta-heuristic SA^D suffers from the extended solution space. Thus, it can be concluded that G2 is a very simple heuristic, which is able to deliver high-quality solutions in negligible time, and, thus, seems well suited for solving the PLP.

The aforementioned tests have exclusively been concerned with computational performance. However, the comparison of G1 and G2 can also be attributed significantly from the warehousing perspective. Recall that G1 represents a typical real-world approach, which is to neglect the joint occurrences of products. Thus, if indeed deterministic order information would be available in a business setting, which is, however, not applied, e.g., due to missing problem awareness, then the gap between both heuristics can be interpreted as the value of order information. The following test series is dedicated to exploring this value.

Typically, the Pareto principle can be observed in many real-world warehouses (e.g., [16,5]), i.e., 80% of the picking effort is caused by only 20% most popular SKUs (fast-moving items). To model this special distribution of picking effort our data generator is adapted as follows. The number of products and orders is invariably set to $n = 100$ and $m = 200$. The additional parameters p and q are introduced, which denote the fraction of most popular items and the fraction of picking effort they cause, respectively. Both parameters are varied as follows: $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $q = \{0.5, 0.6, 0.7, 0.8, 0.9\}$. For any combination of the Pareto values p and q instance generation is repeated $R = 50$ times and any single instance is generated as follows: First, the number f of fast-moving items is set to $f = \lceil p \cdot n \rceil$. Then, for each order its order size is determined by drawing an equally distributed integer number from interval $[1; f]$. Finally, for each order position we draw an equally distributed random number from interval [0; 1]. If this random number is less (larger) than q we add a unique fast-moving (slow-moving) item to the respective order.

The results of this test series are summarized by Fig. 5, where the average gap ($\frac{Z(G1)-Z(G2)}{Z(G2)}$) is depicted in dependency of the Pareto values p and q.

These results suggest the following conclusions with regard to the value of applying order information:

- Whenever there are just a few fast-moving items producing almost the complete picking effort, gathering order information (G2) is only moderately superior to merely applying product specific information (G1). E.g., for $p = 0.1$ and $q = 0.9$ the average gap is only 3.7%. A look on the resulting storage plans for this parameter setting reveals that SKUs of both categories are never mixed, i.e., fast-moving items reside in the front and slow-moving items are assigned to the back of the rack. Clearly, relocating a slow-moving item to the front next to an often jointly ordered fast-mover would induce the placement of another fast-moving item in a less desirable location. Given this bisection, utile relocations due to order information can only occur within each group. However, the group of fast-moving items is small ($p = 0.1$), so that

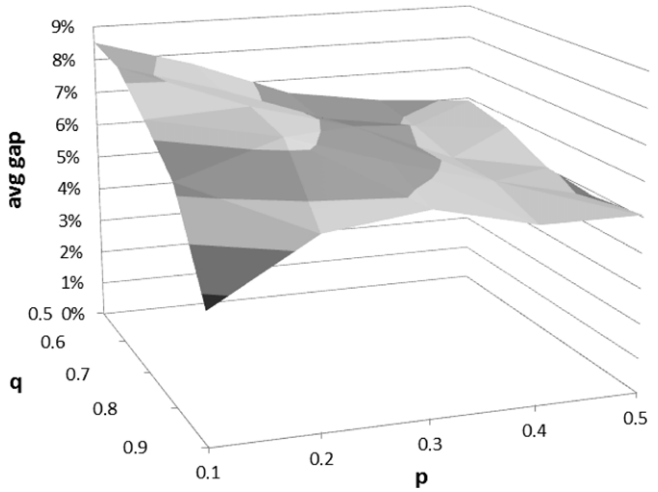


Fig. 5. Average gap between G1 and G2 for varying Pareto values.

there is hardly any flexibility for relocations. On the other hand, ample flexibility within the large group of slow-moving items does not considerably reduce the picking effort, because these items are so rarely ordered ($q = 0.9$).

- A distinct advantage of order information requires either more flexibility in an enlarged fast-moving section, i.e., for $p = 0.5$ and $q = 0.9$ the average gap is 5.1%, or a higher order frequency for slow-moving items, i.e., for $p = 0.1$ and $q = 0.5$ the average gap amounts to 8.5%. Among both cases the latter shows even more advantageous, because better storage plans in the back area of the rack among slow-moving items directly reduce picking effort, whereas relocations in the front area are insignificant for all those orders additionally containing some slow-moving item, so that the fast-movers section needs to be completely passed anyway.
- If the values $p = 0.5$ and $q = 0.5$ are applied all SKUs are equally likely part of each order and the average gap amounts to 5.9%. On the one hand, this dissolves the bisection of items, so that flexibility of relocations over the complete rack exist. On the other hand, the effect of these relocations are not that distinct, because putting one item to the front requires the relocation of another item to the back, which is, however, (nearly equally likely) part of some other orders with, then, enlarged picking effort.

Note that these gains of applying order information constitute merely upper bounds on the actual reduction of picking effort. Typically, order information are not available with certainty, but bound to forecast errors. A further investigation of this coherence is provided in Section 6.

4. Complexity results for other layouts

In this section, we apply the complexity result of our basic PLP described in the previous section for proving NP-hardness of PLPs occurring in other warehouse layouts. Specifically, we consider the PLP with a single rack and a central depot located in the middle of the rack (Section 4.1), the PLP with two parallel racks along a single aisle and a front end depot (Section 4.2), and the PLP with a single two-dimensional rack and a front end depot (Section 4.3).

4.1. Single rack with central depot

The warehouse layout for an PLP with a single straight rack and a central depot being located near to the middle of the rack is depicted in Fig. 6. Again, such a layout may, for instance, occur in a forward area for an efficient picking of fast-moving items (e.g., see [2]). The walking distance per order for a given product assignment can simply be computed by adding the distance to and from the left and right most SKU (provided that an order contains SKUs stored on both sides from the depot). For a more formal description we subdivide the storage locations into subsets L and R located to the left and right of the depot

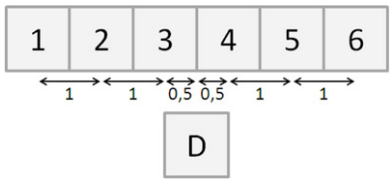


Fig. 6. Single rack with a central depot.

respectively, where a storage position located directly on top of a depot, i.e., for odd number n of storage locations, can be assigned to either set. Without loss of generality we assign such a central storage position to L. For our example of Fig. 6, $L = \{1, 2, 3\}$ and $R = \{4, 5, 6\}$ holds. Furthermore, we introduce a distance vector d, which defines the distance d_s from the depot to each storage position s. Using this notation the total walking distance of a given storage location amounts to: $Z = \sum_{o \in O} \left(2 \cdot \max_{i \in o:\, s(i) \in L}\{d_{s(i)}\} + 2 \cdot \max_{i \in o:\, s(i) \in R}\{d_{s(i)}\}\right)$.

Theorem 2. *The PLP with a single rack and a central depot is NP-hard in the strong sense.*

Proof. We prove this theorem by a reduction from our previous PLP with a single rack and a front end depot. For a better distinction we denote the PLP with a central and front end depot and their data with superscript ^{C} and ^{F}, respectively. For the n^F SKUs and storage positions of PLP^F we introduce three sets P_1^C, P_2^C, and P_3^C of SKUs in PLP^C, where P_1^C and P_2^C each consist of n^F SKUs. Set P_3^C contains two products, so that $n^C = 2n^F + 2$. We, then, transfer the m^F orders of PLP^F to the SKUs of P_1^C without altering the order original structure. Furthermore, we add $Z^F + m^F$ orders each consisting of all SKUs in P_1^C, where Z^F denotes the optimal objective value of the PLP^F instance. Analogously to P_1^C, we extend the order set by another m^F (original) orders plus $Z^F + m^F$ all-item orders referring to the SKUs of P_2^C. Finally, for both items of P_3^C $2 \cdot (Z^F + m^F) \cdot (n^F + 1) + 1$ single-item orders are introduced. Jointly, all three order sets build the order set O^C of PLP^C. The question we ask is whether we can find a solution for PLP^C with objective value $Z^C \leq 8 \cdot (Z^F + m^F) \cdot (n^F + 1) + 2$.

Clearly, both items of P_3^C are to be assigned to the central storage positions closest to the depot, which causes a walking distance of $4 \cdot (Z^F + m^F) \cdot (n^F + 1) + 2$. Any other storage position would cause an objective value higher Z^C. This leaves exactly n^F storage positions to the left and to the right of the depot. Considering each of the $Z^F + m^F$ orders referring to the SKUs in P_1^C and P_2^C, respectively, it is inevitable to assign all SKUs in P_1^C to storage positions to one side of the depot and all SKUs in P_2^C to the other. Otherwise, the remaining walking distance of $Z^C - [4 \cdot (Z^F + m^F) \cdot (n^F + 1) + 2] = 4 \cdot (Z^F + m^F) \cdot (n^F + 1)$ would be exceeded. After an assignment to the left and the right of the depot and the subtraction of the all-item orders, the remaining distance of $2 \cdot (Z^F + m^F)$ has to suffice for each of the m^F orders referring to P_1^C and P_2^C, respectively. The storage locations of these SKUs can directly be transferred from the PLP^F instance without outreaching the remaining objective value, so that the existence of a solution of the PLP^F instance implies the existence of a solution for the PLP^F instance and vice versa, which completes the proof. □

Note that our dynamic programming procedure for our basic PLP with a single rack and a front end depot can simply be adapted to the central depot case. Again, the DP is subdivided into n stages, which are evaluated in a specific order. First, starting from the left most storage position we iterate through the set L of "left" storage locations until reaching the central position. Then, we iterate through set R starting with the right most position. The states in each stage must contain two sets of SKUs assigned to the left and right part of the rack, respectively. As these adoptions are truly straightforward, we abstain from a detailed description.

4.2. Single aisle with front end depot

In many real-world warehouses, the layout consists of multiple aisles each having parallel racks on both sides. We, now, investigate the case of one straight aisle and a front end depot (see Fig. 7). We presuppose that any two storage positions of both racks directly opposing each other have identical distance to (and from) the depot, whereas the distance between both of them is zero. Thus, we assume narrow aisles. For computing the walking distance per order and a given storage assignment we merely have to determine the distance to and from the farthest storage position irrespective of its rack side.

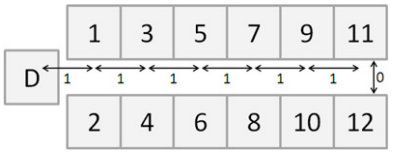


Fig. 7. Single aisle with front end depot.

Theorem 3. *The PLP with one aisle, parallel racks on both sides, and a front end depot (denoted as* PLP^A*) is NP-hard in the strong sense.*

Proof. Again, we prove the theorem by a transformation from the PLP with a single rack and a front end depot (PLP^F). For each SKU $i = 1, \ldots, n^F$ of PLP^F we introduce two SKUs (and, thus, storage positions) i_1 and i_2 within PLP^A. The order set is generated as follows. First, for each SKU $i = 1, \ldots, n^F$ we introduce Z^F orders with $\{i_1, i_2\}$. Additionally, we directly transfer the order set of PLP^F to the SKUs with index 1 within PLP^A. The question we ask is whether we can find a solution for PLP^A with objective value $Z^A \leq Z^F + Z^F \cdot n^F \cdot (n^F + 1)$.

Obviously, any of the n^F SKU pairs are to be assigned to storage positions directly opposite to each other, which causes a walking distance of $Z^F \cdot n^F \cdot (n^F + 1)$ due to the Z^F orders for any pair. Any other assignment would cause an objective value

higher than Z^A. Due to the orders that were directly transferred from the corresponding PLP^F instance the ordering of the pairs directly corresponds to the optimal product locations of the PLP^F instance. Obviously, a feasible ordering exists if and only if the PLP^F instance is a YES-instance. □

Clearly, this proof can easily be extended to show computational complexity for other warehouse settings consisting of multiple aisles. Note that this result is not trivial, because the picker routing subproblems in multi aisle settings are typically special cases of the traveling salesman problem solvable in polynomial time (e.g., see [17,6]), so that NP-hardness for the overall problem does not instantaneously follows from containing an NP-hard routing problem. Further note that solution procedures for PLP^F can directly be transferred to this layout. It only has to be ensured that the storage locations are ordered according to increasing distance from the depot as is depicted in Fig. 7.

4.3. Two-dimensional rack with front end depot

Up to now, only the horizontal distance, i.e., the walking distance of pickers, has been considered. However, with automated cranes (also denoted as storage and retrieval (S/R) machines) multiple shelves arranged on top of each other become accessible. For instance, Van Oudheusden and Zhu [20] consider a person-on-board S/R machine serving a two-dimensional rack with a front end depot (see Fig. 8). Typically, such an S/R machine is propelled with two independent driving mechanisms, so that it is the maximum vertical and horizontal distance (Chebyshev distance or maximum metric), which decides on the actual processing time. We assume that a rack is subdivided into p rows and q columns with horizontal and vertical distances equal to 1. Without loss of generality we assume $p \leq q$. Van Oudheusden and Zhu [20] present a heuristic solution procedure for solving the resulting PLP. Up to now, the complexity status of the problem has been unclear. Theorem 4 will close this gap.

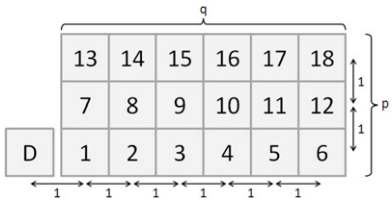


Fig. 8. 2-dimensional rack with a front end depot.

Theorem 4. *The PLP with a two-dimensional rack and a front end depot when operating with the maximum metric (denoted as* PLP^2*) is NP-hard in the strong sense, even if the number p of rows is fixed ($p \neq 2$).*

Proof. Analogously to the previous complexity proofs the transformation is from the basic PLP with a single rack and a front end depot (PLP^F). Since the basic idea of embedding PLP^F into a more complex layout is the same, we only sketch the proof. Given a fixed p and an instance of PLP^F with n^F storage locations (SKUs) and m^F orders we generate the following PLP^2 instance:

- We introduce a rack with p rows and $p + n^F + 1$ columns, so that the number of storage locations (and SKUs) amounts to $n = p \cdot (p + n^F + 1)$.
- Furthermore, we generate $p+2$ sets $N_0 = \{n_1, n_2 \ldots n_p\}$, $N_1, N_2, \ldots, N_p$ and N_{p+1} of SKUs with $|N_0| = p$, $|N_1| = |N_2| = \cdots = |N_p| = n^F$ and $|N_{p+1}| = p^2$. Product set N_1 is the counterpart of the PLP^F products, so that all SKUs of PLP^F are directly transferred to N_1.
- We introduce four order sets O_1, O_2, O_3, O_4:
  O_1: For each SKU in $\bigcup_{j=1}^{p} N_j$ we generate Z^F single-item orders.
  O_2: For any two sets $N_j, N_k \in \{N_1, N_2, \ldots, N_p\}$ we introduce Z^F orders consisting of all SKUs in $N_j \cup N_k \cup \{n_i \in N_0 | j \leq i \leq k\}$.
  O_3: Furthermore, for each SKU in N_{p+1} we introduce $M = M_1 + M_2 + M_3$ single-item orders, where:
  $M_1 = Z^F \cdot p \cdot n^F \cdot (2p + n^F + 1)$,
  $M_2 = Z^F \cdot \left[p \cdot (p-1) \cdot (p + n^F + 1) + \frac{p^3 - p}{6}\right]$, and
  $M_3 = Z^F + 2m^F \cdot p$.
  O_4: Finally, we transfer all orders of PLP^F to set N_1.

The question we ask is whether we can find a solution with objective value $Z^2 \leq 2M \cdot \left[p^3 - \frac{(p-1) \cdot p \cdot (2p-1)}{6}\right] + M$.

Obviously, this transformation can be done in polynomial time. Due to the huge amount of single-item orders in O_3 the SKUs in N_{p+1} have to be assigned to the p^2 storage locations nearest to the depot, which causes a total order picking distance of $2M \cdot \left[p^3 - \frac{(p-1) \cdot p \cdot (2p-1)}{6}\right]$. Any other assignment would worsen the objective value by at least $2M$ and, thus, exceed Z^2.

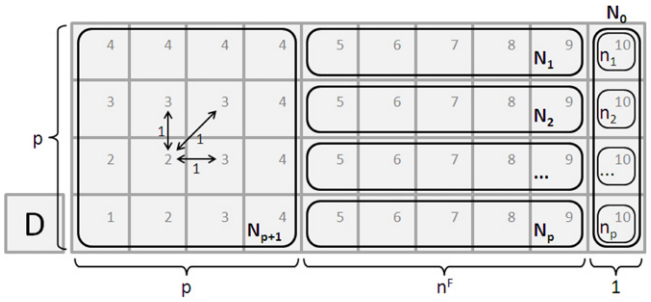


Fig. 9. Assignment of SKUs in $N_0 = \{n_1, \ldots, n_p\}$, $N_1, \ldots, N_{p+1}$ to storage locations.

The remaining SKUs in $N_0, N_1, \ldots, N_p$ have to be assigned to columns $p+1, p+2, \ldots, p+n^F, p+n^F+1$ in such a manner that the cumulated picking distance does not exceed $M = M_1 + M_2 + M_3$. A schematic representation of this assignment pattern is depicted in Fig. 9 with all SKUs in $N_1, N_2 \ldots, N_p$ line by line being assigned to columns $p+1, p+2, \ldots, p+n^F$, all SKUs in N_0 being assigned to column $p+n^F+1$, and the ordering in N_0 corresponding to the ordering of $N_1, N_2, \ldots, N_p$.

Such an assignment yields a total picking distance of exactly M_1 for the orders in O_1, exactly M_2 for the orders in O_2, and at least $2p \cdot m^F$ for the orders in O_4. Note that assigning $N_1, N_2, \ldots, N_p$ to columns $p+1, p+2, \ldots, p+n^F$ is mandatory, because (due to O_1) shifting any of these SKUs to column $p+n^F+1$ would worsen the objective value by $2Z^F$, which cannot be compensated elsewhere. Thus, N_0 has to be assigned to column $p+n^F+1$. Further note that every order in O_2 requires to move up to the last column $p+n^F+1$. Thus, it is necessary to pick an item in every single step on the way from column $p+1$ to $p+n^F+1$, through column $p+n^F+1$ and back to $p+1$. Otherwise, the objective value would be decreased by at least Z^F, which cannot be compensated elsewhere. If $p \geq 3$, the only way to ensure this consists in assigning the SKUs of sets $N_1, \ldots, N_p$ to columns $p+1, \ldots, p+n^F+1$ in such a way that every column contains exactly one item of each set. Vice versa, each set (and especially N_1) is distributed among all columns. Fig. 9 shows one possible assignment. Now it becomes clear, that objective value Z^2 can only be reached if and only if the total distance for picking order set O_4 consisting of products N_1 does not surmount M_3, which in turn is possible if and only if the answer for the corresponding PLP^F instance is YES. □

5. Belt picking with multiple aisles

5.1. Problem description and decomposition approach

In this section, we consider a belt picking environment with multiple aisles, which is a widespread real-world warehouse setting (see [4,21]). We show how our building block solution procedures for the basic layout consisting of a single aisle and a front end depot (see Section 4.2) can readily be applied for solving elementary subproblems in more complex settings. The layout consists of a central conveyor passing multiple aisles (each having two parallel storage racks on both sides serviced by a separate logistics worker) and leading to a sorting and packing depot (see Fig. 10). In such a setting, either order bins travel down the line, into which all SKUs of an order are unified after picking them from their respective storage locations in the racks, or single SKUs are put on the belt and later on unified to orders in the depot (see [4]).

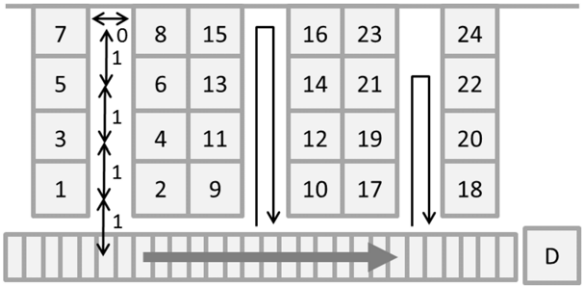


Fig. 10. Warehouse layout for belt picking with multiple aisles.

The resulting PLP has to determine an assignment of $i = 1, \ldots, n$ SKUs to storage locations, which are subdivided into $k = 1, \ldots, K$ sets S_k containing all storage locations of the respective aisle k. With regard to the picker routes it is assumed that in each aisle a picker starts his/her tour at the conveyor belt, collects order positions while walking to the farthest SKU of the current order being stored in his/her aisle, and returns for putting all collected SKUs on the belt. Clearly, if no SKU of the

current order is stored in an aisle no picker tour is required here. Thus, the overall walking distance for a given assignment amounts to:

$$Z = \sum_{o \in O} \sum_{k=1}^{K} 2 \cdot \max_{i \in o} \begin{Bmatrix} d_{s(i)}, & \text{if } s(i) \in S_k \\ 0, & \text{else} \end{Bmatrix}. \tag{4}$$

Under these assumptions, once an assignment of SKUs to aisles is predetermined, the PLP for belt picking (denoted as PLP^B) decomposes into K single aisle PLPs with a front end depot (see Section 4.2). For exploring the performance of such a decomposition approach we implemented a straightforward simulated annealing heuristic, which assigns SKUs to aisles and solves the resulting K PLP^A subproblems with heuristic G2 described in Section 3.3. We compare the resulting solution performance with the optimal results obtained with off-the-shelf solver Xpress and another simulated annealing procedure, which does not follow a problem decomposition and directly assigns SKUs to their dedicated storage positions. Both simulated annealing procedures and the MIP-model are briefly described in the Appendix.

5.2. Computational study

In order to evaluate the performance of the three aforementioned solution procedures the following instances were generated. Different numbers of SKUs ($n \in \{8, 12, 16\}$), orders ($m \in \{0.5 \cdot n, 0.75 \cdot n, n\}$), and aisles K are combined in a full-factorial design, where K is varied from $K = 2$ to $K = \frac{n}{4}$, so that the number of products per aisle $\frac{n}{K}$ is even. Thus, each aisle contains at least four SKUs evenly spread over both racks. Orders of each single instance are generated as is described in Section 3.4. For each parameter setting instance generation is repeated $R = 100$ times, so that in total 1500 instances were obtained.

The results (averaged over all $R = 100$ repetitions and listed in Table 2) show that our decomposition approach SA^A is able to deliver near optimal solutions (with an average gap from the optimum) of less than one percent in reasonable time. In comparison to its competitor, Xpress was not able to obtain optimal results for larger instances, e.g., with $n = 20$ SKUs, within several hours, and SA^D shows higher deviations from the optimum. Thus, it can be concluded that our building block solution procedures for the basic PLP can successfully be applied for solving subproblems in more complex (real-world) warehouse settings.

Table 2
Computational performance for belt picking.

n	m	K	Xpress	SA^A		SA^D	
			CPU time	avg gap	CPU time	avg gap	CPU time
8	4	2	0.02	0.11	5.25	0.00	4.96
	6	2	0.00	0.06	7.27	0.00	6.99
	8	2	0.00	0.14	8.85	0.00	8.48
12	6	2	1.60	0.18	9.69	0.92	9.07
	6	3	1.16	0.00	9.69	0.09	9.11
	9	2	3.75	0.17	13.08	0.84	12.38
	9	3	2.57	0.03	13.08	0.15	12.44
	12	2	6.90	0.19	16.62	0.87	15.69
	12	3	5.94	0.11	16.61	0.35	15.82
16	8	2	115	0.22	15.21	3.55	14.03
	8	4	105	0.24	15.08	1.79	14.27
	12	2	376	0.23	20.60	2.89	19.33
	12	4	464	0.61	20.39	2.02	19.40
	16	2	669	0.19	27.63	2.90	26.18
	16	4	1176	0.81	27.43	2.51	26.45

6. On the robustness of the deterministic product location problem

In this section, we explore the robustness of PLP solutions when executed with incomplete or erroneous input data. Clearly, a problem setting, where the actual order set is known with certainty early enough, so that storage locations can be reorganized, for instance, over night and picked in a distance minimizing manner the next day, seems quite unrealistic. Typically, the order set is merely an anticipated order set bound to forecast errors. To explore the impact of incomplete and erroneous input data the following two additional experiments have been executed.

First, the influence of inaccurately anticipating an order set is explored. We generate an actual order set, i.e., the real order set actually occurring during the planning horizon, and stepwise decrease the randomly drawn sample of orders, i.e., the inaccurately forecasted order set, handed on to the PLP. We determine a storage location plan applying the forecasted order set and evaluate this plan with the actual order set. This way the actual walking distance resulting from a plan determined with incomplete order information and the gap to the best possible solution when solving PLP with the actual order set can be computed. By stepwise reducing the subset of orders handed on to PLP the impact of more or less incompleteness on the solution quality can be explored.

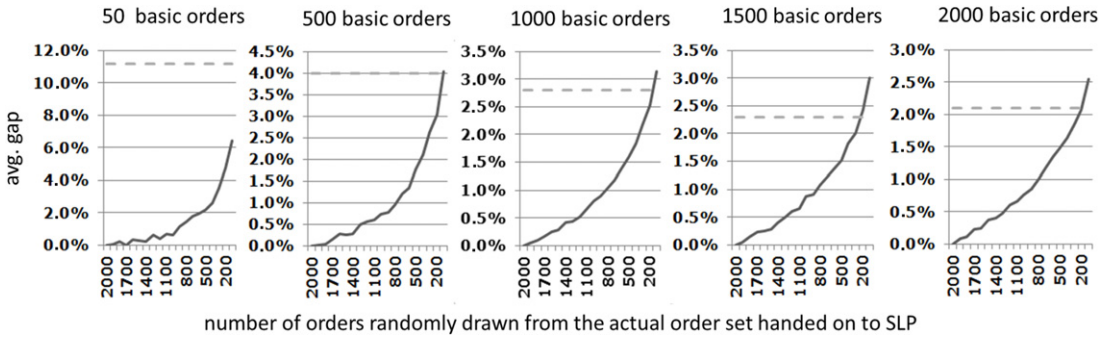


Fig. 11. Impact of incomplete order sets on the performance of PLP.

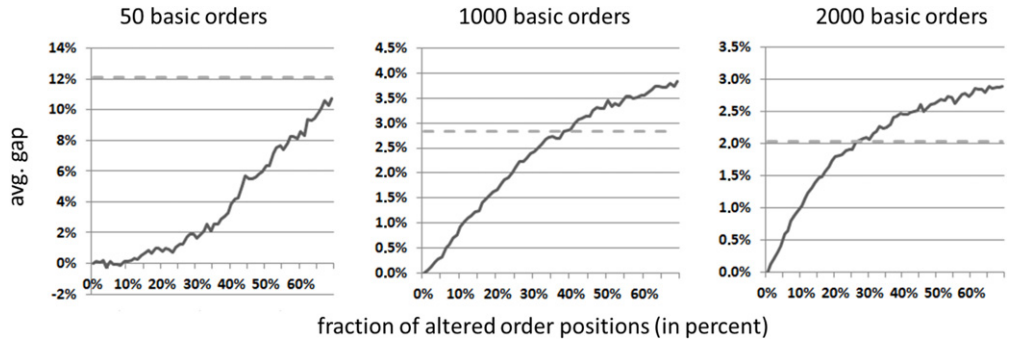


Fig. 12. Impact of erroneous order information on the performance of PLP.

Specifically, the experiment is designed as follows. We investigate the basic layout with a single rack and a front end depot, so that our instance generator of Section 3.4 is applied with $n = 50$ and a varying number of orders ($m \in 50, 500, 1000, 1500, 2000$). Each order in the resulting order set is a basic order for which a random number $0 \leq rnd \leq 1$ is drawn. These basic orders are, then, duplicated, so that their occurrences in a fixed total set of 2000 orders are proportional to the ratios of their respective random numbers *rnd* compared with the total sum of these random numbers. For this actual order set the respective (forecasted) order set handed on to the PLP is determined by randomly reducing the order set to an increasing extend. The resulting storage plans are, then, evaluated with the actual order set, so that the actual walking distance is computed. This actual walking distance is compared to the solution value of PLP executed with the actual order set. The resulting gap averaged over all 100 repetitions per number of basic orders, i.e., $m \in 50, 500, 1000, 1500, 2000$, is denoted as avg. gap within Fig. 11. Note that all PLP instances are solved with our G2 heuristic, since the computational study of Section 3.4 revealed near optimal solutions for G2.

In the first instance, the results summarized in Fig. 11 confirm our expectations: The less complete an order set with which PLP is solved, the larger the gap to the solution value of the perfect forecast. The extent of the gap depends on the number of basic orders. Having only a few basic orders, i.e., $m = 50$, a "wrong" location of an SKU caused by an incomplete and unrepresentative subset of orders leads to a much higher gap than having many basic orders. In the former case, each SKU is contained in only a few frequently occurring basic orders, so that the impact of a "wrong" storage location aggregates over many orders. The dashed line in Fig. 11 depicts the result of our occurrence-based heuristic G1 executed with the actual order set. This benchmark highlights the robustness of deterministic PLP with regard to incomplete data, because even fairly small order samples lead to better results, than neglecting the joint occurrences of SKUs in orders by simply forecasting the total number of occurrences per SKU.

The second experiment does not treat incompleteness of data but erroneous order information. Analogously to the first experiment, initially, the actual order sets are determined for $n = 50$ and a varying number of basic orders $m \in 50, 1000, 2000$, which are then duplicated up to 2000 orders. In this experiment, all 2000 orders are handed on to PLP, but with erroneous information with regard to the composition of each order. For this purpose, we successively alter randomly drawn order positions by changing the drawn SKU to another one, not yet contained in the respective order. By successively altering more and more order positions increasing forecast errors with regard to the composition of orders are emulated. Fig. 12 depicts the average gap (avg. gap) over all 100 instances per parameter setting between the storage plans determined by PLP (with heuristic G2) on the erroneous order set (and its corresponding objective function for the actual order set) and the PLP objective value for the perfect forecast with the actual order set.

Again, our intuition is confirmed: With an increasing number of wrongly anticipated order positions the gap to the perfect forecast widens. Just as well, due to the aggregation effect "wrong" storage locations for SKUs caused by falsely anticipated

order sets turn out as being more severe for a reduced number of basic orders. Comparing the results with the benchmark, i.e., heuristic G1 executed with the actual order set (dashed line), reveals that a considerable percentage of all order positions needs to be altered before neglecting the SKUs' joint order occurrences becomes the better strategy. Again, our deterministic PLP turns out astonishingly robust against erroneous order information.

7. Conclusion

This paper treats the deterministic product location problem (PLP), where products (SKUs) are assigned to storage locations and sequences of locations visited by a picker when retrieving SKUs according to an "out and back" policy are determined. For a given order set, PLP aims to minimize the total order picking distance. Complexity proofs and suited solution procedures for different warehouse layouts are provided and the robustness of PLP solutions when executed with incomplete or erroneous order information is explored.

On the one hand, future research could develop suited exact and heuristic solution procedures for the PLP in a wide variety of possible warehouse layouts. On the other hand, research on the value of order information should be intensified. By evaluating the operational performance of product location plans determined either with order frequencies of isolated SKUs, pair-wise product correlations, or complete order information for representative real-world warehouse layouts, these gains of being better informed could be weighed against the investment cost for improving data availability. This way the practitioner would receive valuable decision support on the right order information gathered and the suited solution method applied for his/her warehouse setting.

Acknowledgments

The authors would like to thank four anonymous referees for their valuable suggestions, which considerably helped to improve the paper.

Appendix A. Simulated annealing procedures for belt picking with multiple aisles

For a better distinction we denote the simulated annealing (SA) procedure only assigning SKU to aisles, so that the problem decomposes into K basic PLP^A problems with a single aisle and a front end depot as SA^A. Our alternative SA, which directly determines the dedicated storage locations of all SKUs, is denoted as SA^D. If both SA procedures are jointly described we simple skip the superscript. Note that the latter procedure SA^D is also applied within Section 3.4 with a single rack.

SA^A and SA^D operate on vectors π^A and π^D with elements $\pi(i)$ for $i = 1, \ldots, n$, which define the aisle $k(i)$ and the dedicated storage position $s(i)$ each SKU i is assigned to, respectively. In order to obtain a neighboring solution both procedures apply a simple swap move, where SA^A (SA^D) randomly draws two aisles (storage positions) interchanging two SKUs. In both cases initial solutions are randomly determined. For a given assignment and SA^D the corresponding objective value $Z^D(\pi)$ can simply be determined by applying Eq. (4). For SA^A solution vector π^A only defines the SKU-aisle assignment, so that K subproblems PLP^A need to be solved by applying heuristic G2 of Section 3.3. For this purpose, each subproblem consists only of the N_k storage positions belonging to aisle k, the set $\{i = 1, \ldots, n : \pi^A(i) = k\}$ of SKUs assigned to k, and a reduced order set, where each actual order only contains those SKUs assigned to aisle k. By summarizing all K results the total walking distance for the respective SKU-aisle assignment is determined.

Whether or not a neighboring solution π' obtained by a swap move is accepted is decided according to traditional probability schemes [1]:

$$\text{Prob}(\pi' \text{ replacing } \pi) = \begin{cases} 1, & \text{if } Z(\pi') \leq Z(\pi) \\ \exp\left(\dfrac{Z(\pi) - Z(\pi')}{C}\right), & \text{otherwise.} \end{cases} \tag{5}$$

If accepted, the current solution π is replaced by π' as the starting point for further local search moves. Our SA is steered by a simple static cooling schedule (see [13]). The initial value for control parameter C is calculated as $Z(\pi^{\text{start}})$, where π^{start} denotes a randomly determined storage location plan. Subsequently, this value C is continuously decreased in the course of the procedure by multiplying it with factor 0.995 in each iteration. A total of 10,000 solutions is evaluated by each of our SA-approaches and the solution with the minimum objective function value $Z(\pi)$ is returned. In our computational study, we have only used control parameter values as described above. Note that preliminary studies have indicated that this parameter setting outperforms other settings and has obtained promising results.

Appendix B. Mixed integer PLP-model for belt picking with multiple aisles

With the notation summarized in Table 3 the PLP in a belt picking environment can be formalized as a MIP-model by objective function (6) and constraints (7)–(10):

$$\text{PLP}^B: \text{Minimize } Z(X, Y) = \sum_{o \in O} \sum_{k=1}^{K} 2 \cdot y_{ok} \tag{6}$$

Table 3
Notation.

P	Set of all SKUs (index i)
K	Number of aisles (and pickers) (index k)
S_k	Set of storage positions belonging to aisle k (index j)
O	Set of orders (index o)
G_o	Set of SKUs contained in order o
d_j	Distance to and from storage position j from the depot
x_{ij}	Binary variable: 1, if SKU i is assigned to storage position j, 0 otherwise
y_{ok}	Continuous variable: walking distance in aisle k for picking order o

subject to

$$\sum_{k=1}^{K} \sum_{j \in S_k} x_{ij} = 1 \quad \forall\, i \in P \tag{7}$$

$$\sum_{i \in P} x_{ij} = 1 \quad \forall\, k = 1, \ldots, K;\ j \in S_k \tag{8}$$

$$y_{ok} \geq d_j \cdot x_{ij} \quad \forall\, o \in O;\ i \in G_o;\ k = 1, \ldots, K;\ j \in S_k \tag{9}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, i \in P;\ k = 1, \ldots, K;\ j \in S_k. \tag{10}$$

Objective function (6) minimizes the total walking distance. Constraints (7) and (8) ensure a one-to-one mapping between SKUs and storage positions and constraints (9) force variables y_{ok} to the maximum distance d_j of the storage locations $j \in S_k$ being located in aisle k and assigned to an SKU $i \in G_o$ belonging to current order o.

References

[1] E.H.L. Aarts, J.H.M. Korst, J.M. van Laarhoven, Simulated annealing, in: E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Chichester, 1997, pp. 91–120.
[2] J.J. Bartholdi, S.T. Hackman, Allocating space in a forward pick area of a distribution center for small parts, IIE Transactions 40 (2008) 1046–1053.
[3] H. Brynzér, M.I. Johansson, Storage location assignment: using the product structure to reduce order picking times, International Journal of Production Economics 46 (1996) 595–603.
[4] R. de Koster, Performance approximation of pick-to-belt orderpicking systems, European Journal of Operational Research 72 (1994) 558–573.
[5] R. de Koster, T. Le-Duc, J.K. Roodbergen, Design and control of warehouse order picking: a literature review, European Journal of Operational Research 182 (2007) 481–501.
[6] R. de Koster, E.S. van der Poort, Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions, IIE Transactions 30 (1998) 469–480.
[7] E.A. Frazelle, G.P. Sharp, Correlated assignment strategy can improve order-picking operation, Industrial Engineering 4 (1989) 33–37.
[8] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[9] J.X. Gu, M. Goetschalckx, L.F. McGinnis, Research on warehouse operation: a comprehensive review, European Journal of Operational Research 177 (2007) 1–21.
[10] J.X. Gu, M. Goetschalckx, L.F. McGinnis, Research on warehouse design and performance evaluation: a comprehensive review, European Journal of Operational Research 203 (2010) 539–549.
[11] M. Held, R.M. Karp, A dynamic programming approach to sequencing problems, Journal of the Society for Industrial and Applied Mathematics 10 (1962) 196–210.
[12] J.L. Heskett, Cube-per-order index—a key to warehouse stock location, Transportation and Distribution Management 3 (1963) 27–31.
[13] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[14] P. Kouvelis, W. Chiang, G. Yu, Optimal algorithms for row layout problems in automated manufacturing systems, IIE Transactions 27 (1995) 99–104.
[15] G. Palubeckis, A branch-and-bound algorithm for the single-row equidistant facility layout problem, OR Spectrum 34 (2012) 1–21.
[16] C.G. Petersen, An evaluation of order picking policies for mail order companies, Production and Operations Management 9 (2000) 319–335.
[17] H.D. Ratliff, A.S. Rosenthal, Orderpicking in a rectangular warehouse: a solvable case of the traveling salesman problem, Operations Research 31 (1983) 507–521.
[18] J. Renaud, A. Ruiz, Improving product location and order picking activities in a distribution centre, Journal of the Operational Research Society 59 (2008) 1603–1613.
[19] K.J. Roodbergen, I.F.A. Vis, A survey of literature on automated storage and retrieval systems, European Journal of Operational Research 194 (2009) 343–362.
[20] D.L. Van Oudheusden, W. Zhu, Storage layout of AS/RS racks based on recurrent orders, European Journal of Operational Research 58 (1992) 48–56.
[21] M. Yu, R. de Koster, Performance approximation and design of pick-and-pass order picking systems, IIE Transactions 40 (2008) 1054–1069.