Journal of Scheduling
https://doi.org/10.1007/s10951-024-00811-2

A polynomial-time dynamic programming algorithm for an optimal picking problem in automated warehouses

Michele Barbato[1] · **Alberto Ceselli**[1] · **Giovanni Righini**[1]

Accepted: 29 April 2024

Abstract
We consider an optimization problem arising when a set of items must be selected and picked up from given locations in an automated storage and retrieval system by a crane of given capacity, minimizing the overall distance traveled. The problem has been classified as open in a recent taxonomy of optimal picking problems in automated warehouses. In this paper, we analyze some non-trivial properties of the problem and we describe a polynomial-time dynamic programming algorithm to solve it to proven optimality.

Keywords Combinatorial optimization · Crane scheduling · Polynomial-time algorithm · Dynamic programming

1 Introduction

Automated storage and retrieval systems (AS/RS) are commonly used to speed up warehouse operations in industrial production plants (Azadeh, De Koster, and Roy, 2019). At their core, a crane operates to pickup boxes at specific positions in the warehouse shelves and to bring them to an output port, where a conveyor belt is placed. To be efficient in terms of reactiveness and energy consumption, the crane schedule must be suitably optimized. This is why an appropriate modeling and the design of efficient algorithms are crucial and why crane scheduling is a lively topic in mathematical optimization, aiming at the development of exact algorithms (Weidinger, Boysen, and Schneider, 2019) as well as heuristics (Fontin & Lin, 2020; Ouzidan, Sevaux, Olteanu, Pardo, and Duarte, 2022). For an overview of AS/RS systems and their optimization we refer the reader to surveys, such as van den Berg and Zijm (1999); Gu et al. (2007); de Koster et al. (2007); Roodbergen and Vis (2009); Boysen et al. (2019). In this paper we consider an automated warehouse where all components are stored in identical (standardized) boxes, called *items* in the remainder. The items are stored in suitable locations along a single aisle. All the required items are collected by a single crane equipped with q shuttles, which moves along a rail. The crane is initially empty at an idle point that also acts as the output location. The idle point (called *origin* in the remainder) is at a given intermediate position along the rail. Such a layout appears in real-world warehousing systems studied in the literature, see, e.g., Tang and Ren (2010). Therefore, the aisle can be represented by two lines with a common origin, as in Fig. 1. In general, each required item can be picked up on either line, because multiple identical copies of the same materials or components can be stored in the automated warehouse. We consider the case where each required item can be picked up from two distinct locations, one on each line. This assumption does not imply any loss of generality: in case an item is available at more than one location on a given line, it is always optimal to pick it up from the location that is closest to the origin; in case an item is not available on a given line, it can be considered as if it were located at a very large distance from the origin on that line. At each cycle, the crane starts from the origin, it moves along one of the two lines, it collects at most q required items and it returns to the origin where it unloads the collected items. Hence, the total distance traveled in a cycle is twice the distance between the origin and the far-

M. Barbato, A. Ceselli and G. Righini have contributed equally to this work.

✉ Michele Barbato
michele.barbato@unimi.it

Alberto Ceselli
alberto.ceselli@unimi.it

Giovanni Righini
giovanni.righini@unimi.it

[1] Dipartimento di Informatica "Giovanni degli Antoni", Università degli Studi di Milano, Via Celoria 18, 20133 Milano, Italy

Published online: 16 July 2024

thest collected item. The objective is to minimize the total distance traveled by a crane of capacity q to collect a set of required items.

The problem described above is a case of single-crane scheduling (Boysen & Stephan, 2016; Goeke & Schneider, 2021), and in particular it can be seen as a variant of $F, q/IO^1/C_{\max}$: single IO point with crane composed by q shuttles, single command (only pickup operations), makespan minimization. Still in the context of single-crane scheduling, the paper by Barbato et al. (2019) discusses several pickup and delivery problems arising in the study of a real-world automated warehouse and studies their worst-case time complexity. The real-world system considered in Barbato et al. (2019) has a storage area where each cell of a 2-dimensional rack has depth 2. The items are collected by a crane of capacity 2. The storage area communicates with a production area by means of an automated belt, from which items are delivered to human operators by means of shuttle of capacity 2. Barbato et al. (2019) focuses on the worst-case time complexity of the problems arising in this context, as a function of the warehouse characteristics (e.g., capacity of the vehicles, number of dimensions of the storage area, etc.). The same paper also introduces a four-field notation to classify them. Our problem is accordingly denoted as $q/1/P/V$. Such a notation indicates that an arbitrary capacity value is given (field "q"), that the warehouse is 1-dimensional (field "1") and only pickup operations (field "P") must be performed to collect items from locations to be selected (field "V"). The assessment of the time complexity of $q/1/P/V$ had been left open in Barbato et al. (2019) and, to the best of our knowledge, no other results in the literature can be found regarding $q/1/P/V$. As reported in Boysen and Stephan (2016), several single-crane scheduling problems have been proven **NP**-hard for arbitrary capacities (i.e., cases $F, q/\cdot/\cdot$ of Boysen and Stephan (2016)). The combinatorial structure of problems with capacity 1 has instead been exploited to design polynomial-time exact algorithms (Gharehgozli, Yu, Zhang, and Koster, 2017). A few special cases exist, which can be solved in polynomial time also when the capacity is 2 (see, e.g., Dooly and Lee (2008)). These cases, however, do not directly extend to the setting considered in this paper. Several problems involving capacity 3 (e.g. $F, 3/\cdot/\cdot$ of Boysen and Stephan (2016)) are NP-hard. Although problem $q/1/P/V$ has been inspired by the study of a real-world AS/RS, it has no direct practical applications. Our main interest in this problem is rather motivated by its relevance as an intermediate step to solve more complex problems in real-world systems as the one described in Barbato et al. (2019).

In this paper, we provide a polynomial-time algorithm to solve $q/1/P/V$. Our result holds for any value of q, making it a non-trivial methodological achievement in the literature.

Outline.

The core of our approach is the identification of dominance properties allowing to restrict the search to a specific set of structured solutions and hence to avoid the combinatorial explosion in the number of solutions to be examined. In Sect. 2, we formally define the problem $q/1/P/V$ and observe that it is decomposable into two sub-problems: determining a line assignment of items and optimally collecting the items on the assigned line. We state the dominance property that allows to restrict the set of solutions for each given line assignment. Next, Sect. 3 presents dominance properties that allow to restrict the set of line assignments to be enumerated. Finally, these dominance properties are exploited in the dynamic programming algorithm to solve $q/1/P/V$, outlined in Sect. 4. The conclusions of our paper are given in Sect. 5.

2 Problem definition and decomposition

Throughout this paper, we indicate by $N = \{1, 2, \ldots, n\}$ the set of items to be picked up from the 1-dimensional warehouse described in the introduction. The origin is indicated by O. Each line holds one copy of each item $i \in N$ at a given location. From now on, $d_\ell(i)$ denotes the distance of $i \in N$ from O on line $\ell \in \{1, 2\}$. Multiple items can be stored at a same location. The crane capacity is a given positive integer q. An illustration of the above definitions in a simple case with $|N| = 2$ and $q = 2$ is provided in Fig. 1.

Definition 1 (Trips) A *trip* T is a subset of N of cardinality at most q. The *cost* of a trip T on line $\ell \in \{1, 2\}$ is $C_\ell(T) = \max_{i \in T}\{d_\ell(i)\}$, i.e., half the distance traveled by the crane.

The problem $q/1/P/V$ requires to find a pair $(\mathcal{T}^1, \mathcal{T}^2)$ of sets of non-empty trips such that the trips in $\mathcal{T}^1 \cup \mathcal{T}^2$ partition N and such that the total cost $C(\mathcal{T}) = \sum_{\ell=1}^{2} \sum_{T \in \mathcal{T}^\ell} C_\ell(T)$ is minimum.

Definition 2 (Leading item) A *leading item* of a trip T on line ℓ is an item in T that is farthest from O, i.e., an item $j \in T$ such that $d_\ell(j) = \max_{i \in T}\{d_\ell(i)\}$. A leading item is not necessarily unique.

From the previous definition, the cost of each trip T on line ℓ is $C_\ell(T) = d_\ell(j)$, where j is a leading item of T.

Determining a feasible solution to $q/1/P/V$ consists of (a) deciding a *line assignment*, i.e., determining the line where each item must be picked up and (b) grouping the items assigned to the same line into trips.

Optimally grouping items on each line.

We first consider the sub-problem of optimally grouping the items consistently with a given line assignment $\mathcal{A}$. In this sub-problem, the locations where items are to be collected are fixed. Following the notation introduced in Barbato et al.

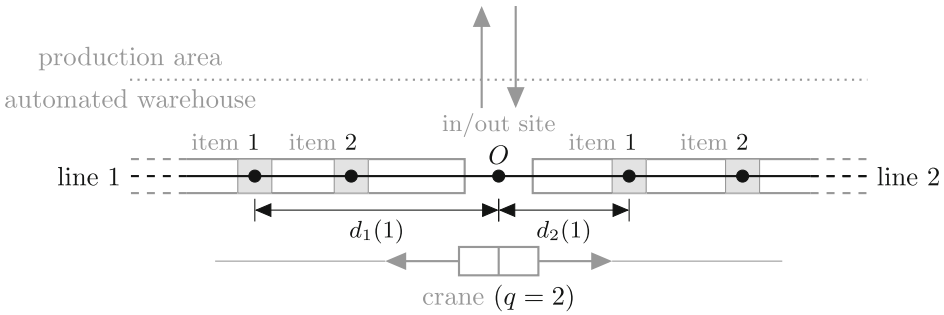


Fig. 1 Schematic top view of the 1-dimensional automated warehouse considered in this paper. The real elements (crane, rail, item boxes, in/out site) are depicted in gray, while the formal representation of the warehouse (consisting in the origin O, lines and points corresponding to the item locations) is drawn in black. Each line holds a copy of an element in $N = \{1, 2\}$. The distances $d_1(1)$ and $d_2(1)$ of item 1 on each line are explicitly represented

(2019) and emphasizing the dependency on $\mathcal{A}$, we denote this sub-problem as $q/1/P/F(\mathcal{A})$. Clearly, $q/1/P/F(\mathcal{A})$ corresponds to two independent instances, one for each line, of the problem with fixed locations on a single line.

Definition 3 ($q/1/P/F$ on a single line) **Data:**

- a set N of n locations to be visited on a line,
- the distance $d(i)$ from the origin for each item $i \in N$,
- the crane capacity q.

Constraints: find a partition of N into a set $\mathcal{T}$ of trips such that $|T| \leq q \ \forall T \in \mathcal{T}$.
Objective: minimize the total cost of the trips, $C(\mathcal{T})$.

The problem of Def. 3 can be solved to optimality by the greedy algorithm described by Brucker et al. (1998) with worst-case time complexity $O(n \log n)$: on each line, items are sorted according to their distance from O and they are grouped in batches of q starting from the farthest ones.

Owing to this optimal greedy algorithm, the implicit complete enumeration of solutions can obtained through the implicit complete enumeration of line assignments.

The solution computed by the greedy algorithm may not be unique, because some items may be located at the same distance from O. However, it is possible to characterize the whole set of optimal solutions by two properties, outlined hereafter.

For any given line $\ell \in \{1, 2\}$ and any given set N_ℓ of items assigned to it, let $\mathcal{T}^\ell = \{T_1^\ell, T_2^\ell, \ldots, T_m^\ell\}$ a set of trips partitioning N_ℓ.

Definition 4 (Compact trips and solutions) A set $\mathcal{T}^\ell$ of trips on line $\ell \in \{1, 2\}$ is *compact* if and only if, for any two distinct trips $T_1, T_2 \in \mathcal{T}^\ell$, either $d_\ell(i) \geq d_\ell(j) \ \forall i \in T_1, \forall j \in T_2$ or $d_\ell(i) \leq d_\ell(j) \ \forall i \in T_1, \forall j \in T_2$. A solution $(\mathcal{T}^1, \mathcal{T}^2)$ to $q/1/P/V$ is *compact* if and only if $\mathcal{T}^\ell$ is compact for any line $\ell \in \{1, 2\}$.

Definition 5 (Complete trips and solutions) Consider a set $\mathcal{T}^\ell$ of m_ℓ trips on line $\ell \in \{1, 2\}$ ordered by non-increasing distance of their leading items. $\mathcal{T}^\ell$ is *complete* if and only if its farthest $m_\ell - 1$ trips are made of q items each. A feasible solution $(\mathcal{T}^1, \mathcal{T}^2)$ to $q/1/P/V$ is *complete* if and only $\mathcal{T}^\ell$ is complete for $\ell = 1, 2$.

In Fig. 2, we illustrate the concepts of compactness and completeness by means of simple examples.

Observation 1 By construction, the optimal solution to $q/1/P/F(\mathcal{A})$, computed by the greedy algorithm of Brucker et al. (1998), is compact and complete.

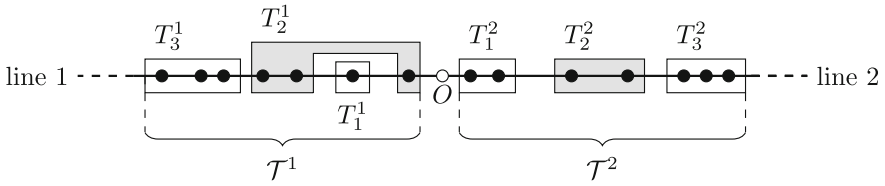
The converse is also true.

Property 1 (Optimality) *For any given line assignment $\mathcal{A}$, any solution that is compact and complete is optimal for $q/1/P/F(\mathcal{A})$.*

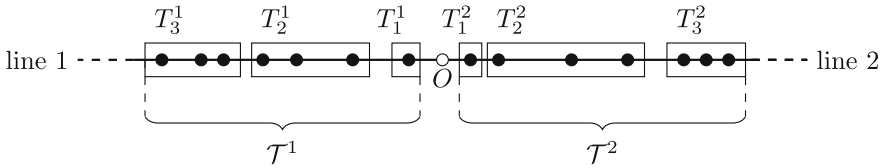
Proof By definition, two complete solutions to a $q/1/P/F(\mathcal{A})$ instance have the same number of trips, which can be put in correspondence, after sorting them by their cost. If the two solutions are also compact, the leading items of two corresponding trips appear in the same position in N_ℓ. Then all compact and complete solutions have the same cost and are optimal by Observation 1. □

The latter property allows to compare *partial* line assignments, to early discard some that cannot correspond to optimal solutions, as described in Sect. 3.

Graphical representation of line assignments.
For the sake of illustrating the dominance properties between line assignments and the algorithm that relies upon them, we introduce a graph terminology, associating each item in N with an *edge*. This terminology stems from representing the two lines of the warehouse as vertical and parallel and each item as an edge linking its positions on the two lines as in Fig. 3a. Hence, in the remainder the terms *item* and *edge* will be used as synonymous.



(a) For $q = 3$, the set of left-hand trips $\mathcal{T}^1$ is complete but it is not compact because T_2^1 and T_1^1 overlap; the set of right-hand trips $\mathcal{T}^2$ is compact but not complete because T_2^2 collects two items. The gray trips are those violating the definitions of compactness and completeness.



(b) For $q = 3$, both $\mathcal{T}^1$ and $\mathcal{T}^2$ are compact and complete, hence the solution $(\mathcal{T}^1, \mathcal{T}^2)$ also is compact and complete.

Fig. 2 Examples of a non-compact and non-complete solution (top) and a compact and complete solution (bottom) for $q = 3$

Definition 6 (Orientations) Edge $i \in N$ is *horizontal* if and only if $d_1(i) = d_2(i)$. In a given line assignment, a horizontal edge is ℓ-*oriented* if and only if item i is assigned to line ℓ; a non-horizontal edge $i \in N$ is *upward-oriented* if and only if item i is assigned to the line where it is farther from the origin and it is *downward-oriented* if and only if it is assigned to the line where it is closer to the origin.

Assigning an item to one of the lines corresponds to orienting its edge, as shown in Fig. 3b.

Definition 7 (Intersecting edges) Given two distinct edges $i \in N$ and $j \in N$, they *intersect* if and only if $d_{\ell'}(i) \leq d_{\ell'}(j)$, $d_{\ell''}(i) \geq d_{\ell''}(j)$ where $\{\ell', \ell''\} = \{1, 2\}$. Two distinct edges are *disjoint* if and only if they do not intersect.

3 Dominance properties for line assignments

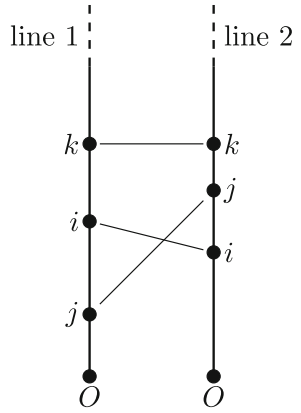
Since there are n items to be assigned in a problem instance, there are 2^n possible line assignments. To contrast the combinatorial explosion implied by the exponential number of line assignments, the dynamic programming algorithm illustrated in Sect. 4 restricts the search to a subset of canonical line assignments corresponding to non-dominated solutions to $q/1/P/V$. Such "non-dominated" line assignment are defined in Subsection 3.1. Then, in Subsection 3.2 we introduce the definition of *primary edge sets*, and we show that they are in correspondence with non-dominated line assignments. Finally, in Subsection 3.3 we study the properties of *partial line assignments*, which are iteratively constructed by selecting primary edges according to a suitably defined partial order of the edges. In particular, we establish dominance properties between partial line assignments, which are the basis for a polynomial-time dynamic programming algorithm, since they allow discarding some sub-optimal partial line assignments.

3.1 Non-dominated line assignments

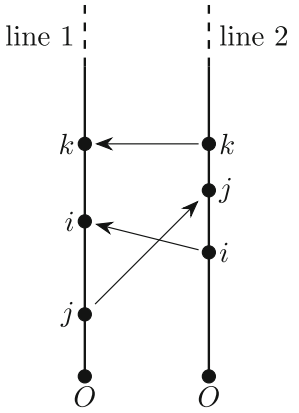
The dominance relation between line assignments relies on some definitions and properties presented hereafter. First, we show the effect of replacing an item on a line: it is easy to characterize replacements that do not produce a cost increase on the line, independently of the position of all the other items. Then, we apply this observation to show the effect of swapping the assignments of two items: we characterize swaps that do not worsen solutions, independently of the assignments of all the other items. Then, we introduce a lexicographic ordering of line assignments that allows to define an asymmetric and transitive dominance relation between them. Finally, we prove that an optimal solution is certainly found even if the search is restricted to line assignments that are non-dominated according to our definition of dominance.

Property 2 (Replacement) *Consider a line $\ell \in \{1, 2\}$ and a compact and complete set $\mathcal{T}^\ell$ of trips on that line; let C be its cost. If an item $i \in \mathcal{T}^\ell$ is replaced by an item $j \notin \mathcal{T}^\ell$ with $d_\ell(j) \leq d_\ell(i)$, then the cost of a compact and complete set of trips collecting the items in $\mathcal{T}^\ell \setminus \{i\} \cup \{j\}$ on line ℓ is not larger than C.*

Proof Consider the effect of inserting j in the same trip $T \in \mathcal{T}^\ell$ of the deleted item i. Let $C(T)$ and $C'(T)$ be the cost of trip T before and after the replacement.



(a) Edges associated with items. Item k is horizontal.



(b) A line assignment of the items in Fig. 3a. It corresponds to an orientation of the corresponding edges: the head of each edge indicates the assigned line.

Fig. 3 Edges and orientations representing item positions and a line assignment

If i is not a leading item in T and j does not become the leading item, then $C'(T) = C(T)$.

If i is not a leading item in T and j becomes a leading item, then there exists an item $k \in T$ such that $C(T) = d_\ell(k) \geq d_\ell(i)$ and $C'(T) = d_\ell(j) \geq d_\ell(k)$. Since $d_\ell(i) \geq d_\ell(j)$, then $d_\ell(j) = d_\ell(k) = d_\ell(i)$. Hence, $C'(T) = C(T)$.

If i is a leading item of T and j does not become a leading item, there exists an item $k \in T$ such that $C'(T) = d_\ell(k)$ and $C(T) = d_\ell(i) \geq d_\ell(k)$. Hence, $C'(T) \leq C(T)$.

If i is a leading item of T and j becomes a leading item, then $C(T) = d_\ell(i)$ and $C'(T) = d_\ell(j)$. Since $d_\ell(j) \leq d_\ell(i)$, then $C'(T) \leq C(T)$.

In general, the replacement may produce a set of trips $\mathcal{T}'$ on line ℓ that is not compact and complete. However, owing to Property 1, the cost of a compact and complete set of trips on line ℓ is not larger than the cost of $\mathcal{T}'$. □

Applying Property 2 to both lines immediately yields the following:

Property 3 (Swap) *Consider a solution with a compact and complete set $\mathcal{T}^\ell$ of trips on each line $\ell \in \{1, 2\}$ and let C be its cost. If two items $i \in \mathcal{T}^1$ and $j \in \mathcal{T}^2$ with $d_1(j) \leq d_1(i)$ and $d_2(i) \leq d_2(j)$ are swapped, then the cost of the optimal solution corresponding to the new assignment is not larger than C.*

A graphical representation of Property 3 is illustrated in Fig. 3b, where edges i and j intersect each other: reversing their orientation assigns the same number of items to each line without increasing their distance from O. Hence, the compact solution obtained from the line assignment of Fig. 3b is dominated (in terms of traveled distance) by the compact solution obtained by the line assignment in which edges i and j have opposite orientation while all the other items (not represented in the figure) keep their orientation.

Exploiting Property 3, we now establish a dominance relation between line assignments. For this purpose, we introduce three quantities α, β and γ associated with line assignments, in order to break ties. For a given line assignment $\mathcal{A}$, we indicate by $L(i, \mathcal{A}) \in \{1, 2\}$ the line to which item $i \in N$ is assigned in $\mathcal{A}$.

Definition 8 (α,β,γ) For any given line assignment $\mathcal{A}$

$$\alpha(\mathcal{A}) = \sum_{i \in N} d_{L(i,\mathcal{A})}(i),$$
$$\beta(\mathcal{A}) = \sum_{i \in N} d_{L(i,\mathcal{A})}(i)i,$$
$$\gamma(\mathcal{A}) = \sum_{i \in N} L(i, \mathcal{A})i.$$

Example 1 Consider $N = \{1, 2, 3\}$ with $d_1(1) = 5 = d_2(2)$, $d_1(2) = 3 = d_2(3)$ and $d_1(3) = 4 = d_2(1)$. Let $\mathcal{A}$ be the assignment given by $L(1, \mathcal{A}) = 1$, $L(2, \mathcal{A}) = 2$ and $L(3, \mathcal{A}) = 1$. Then $\alpha(\mathcal{A}) = 14$, $\beta(\mathcal{A}) = 27$ and $\gamma(\mathcal{A}) = 8$.

Property 4 (Edge reversal) *Let $\{\ell', \ell''\} = \{1, 2\}$. Consider two distinct intersecting edges $i \in N$ and $j \in N$.*

1. *Non-coincident edges. If $d_{\ell'}(i) \geq d_{\ell'}(j)$, $d_{\ell''}(i) \leq d_{\ell''}(j)$ and at least one of the two inequalities is strict, consider a line assignment $\mathcal{A}$ in which i is oriented to ℓ' and j is oriented to ℓ'' and the line assignment $\overline{\mathcal{A}}$ obtained from $\mathcal{A}$ by reversing the orientation of both edges. Then, $C(\overline{\mathcal{A}}) \leq C(\mathcal{A})$. Furthermore $\alpha(\overline{\mathcal{A}}) < \alpha(\mathcal{A})$.*
2. *Coincident non-horizontal edges. If $d_{\ell'}(i) = d_{\ell'}(j) > d_{\ell''}(i) = d_{\ell''}(j)$ and $i > j$, consider a line assignment*

$\mathcal{A}$ in which i is oriented to line ℓ' and j is oriented to line ℓ'' and the line assignment $\overline{\mathcal{A}}$ obtained from $\mathcal{A}$ by reversing the orientation of both edges. Then, $C(\overline{\mathcal{A}}) = C(\mathcal{A})$. Furthermore $\alpha(\overline{\mathcal{A}}) = \alpha(\mathcal{A})$ and $\beta(\overline{\mathcal{A}}) < \beta(\mathcal{A})$.

3. *Coincident horizontal edges. If $d_1(i) = d_1(j) = d_2(i) = d_2(j)$ and $i > j$, consider a line assignment $\mathcal{A}$ in which i is oriented to line 2 and j is oriented to line 1 and the line assignment $\overline{\mathcal{A}}$ obtained from $\mathcal{A}$ by reversing the orientation of both edges. Then, $C(\overline{\mathcal{A}}) = C(\mathcal{A})$. Furthermore, $\alpha(\overline{\mathcal{A}}) = \alpha(\mathcal{A})$, $\beta(\overline{\mathcal{A}}) = \beta(\mathcal{A})$ and $\gamma(\overline{\mathcal{A}}) < \gamma(\mathcal{A})$.*

Proof In all three cases, the number of items assigned to each line is the same in $\mathcal{A}$ and $\overline{\mathcal{A}}$. In case 1 reversing the orientation of edges i and j has the only effect of replacing an item with another one at a non-larger distance on each line. This cannot increase the total cost of an optimal solution to $q/1/P/F(\mathcal{A})$, owing to Property 3. The strict decrease in the value of α directly comes from the definition of α and the assumption that at least one of the two inequalities is strict. In case 2 the proof that $C(\mathcal{A})$ and the value of α remain unchanged is trivial. The strict decrease in the value of β comes from the definition of β and the assumption that $i > j$. In case 3, the proof that $C(\mathcal{A})$ and the value of α and β remain unchanged is trivial. The strict decrease in the value of γ comes from the definition of γ and the assumption that $i > j$. □

Definition 9 (Dominance) Given two line assignments $\mathcal{A}$ and $\mathcal{A}'$, $\mathcal{A}$ *dominates* $\mathcal{A}'$ if and only if $\mathcal{A}$ is obtained from $\mathcal{A}'$ through a sequence of the edge-reversal operations described in Property 4.

Corollary 5 *The dominance relation of Def. 9 is asymmetric and transitive.*

Proof We associate with each line assignment $\mathcal{A}$ the triple $(\alpha(\mathcal{A}), \beta(\mathcal{A}), \gamma(\mathcal{A}))$. If $\mathcal{A}$ dominates $\mathcal{A}'$, then, by Property 4, the triple $(\alpha(\mathcal{A}), \beta(\mathcal{A}), \gamma(\mathcal{A}))$ is lexicographically smaller than $(\alpha(\mathcal{A}'), \beta(\mathcal{A}'), \gamma(\mathcal{A}'))$, i.e., $(\alpha(\mathcal{A}) < \alpha(\mathcal{A}')) \vee ((\alpha(\mathcal{A}) = \alpha(\mathcal{A}')) \wedge (\beta(\mathcal{A}) < \beta(\mathcal{A}'))) \vee ((\alpha(\mathcal{A}) = \alpha(\mathcal{A}')) \wedge (\beta(\mathcal{A}) = \beta(\mathcal{A}')) \wedge (\gamma(\mathcal{A}) < \gamma(\mathcal{A}')))$. Since the lexicographical order is asymmetric and transitive so is the dominance relation of Def. 9. □

Corollary 6 *Every instance of $q/1/P/V$ admits at least one optimal solution whose corresponding line assignment is non-dominated.*

Proof Let $\mathcal{A}'$ be an optimal line assignment, i.e., the line assignment corresponding to an optimal solution. Assume $\mathcal{A}'$ is dominated by $\mathcal{A}$, as otherwise the result follows immediately. By Property 4 $C(\mathcal{A}) \leq C(\mathcal{A}')$, hence also $\mathcal{A}$ is optimal. Since the dominance relation is transitive and asymmetric, and since the optimal line assignments are finite, the result follows. □

This allows to restrict the search for an optimal solution by considering only non-dominated line assignments.

3.2 Primary sets

So far, we have proven that the search for an optimal solution can be pursued by enumerating line assignments that are non-dominated according to a suitable definition of dominance. The implicit complete enumeration of non-dominated line assignments must be done efficiently. For this purpose, we introduce the definitions of primary edges and primary set and we prove that enumerating primary sets is equivalent to enumerating non-dominated line assignments. While a line assignment requires to specify the orientation of each of the n edges, a primary set is described by a selection of a (typically small) subset of the n edges. We prove that the selection of the primary edges is enough to determine the orientation of all edges in a non-dominated line assignment.

The orientation of the edges relies on the properties of non-dominated line assignments and the definition of primary edges. In turn, the definition of primary edges relies upon the definition of implications between edges. Therefore, this subsection is organized in three paragraphs, one for each of these steps.

Implication between edges.

Definition 10 (Implication between non-horizontal edges) Given two distinct edges $i \in N$ and $j \in N$, with $d_{\ell'}(i) < d_{\ell''}(i)$, *$i$ implies j* if and only if the following three conditions are satisfied:

1. $d_{\ell'}(j) \geq d_{\ell'}(i)$,
2. $d_{\ell''}(j) \leq d_{\ell''}(i)$,
3. at least one of the two inequalities above is strict or $j < i$.

Implication between non-horizontal edges is shown in Fig. 4.

Definition 11 (Implication between horizontal edges) Given two distinct edges $i \in N$ and $j \in N$, with $d_1(i) = d_2(i)$, *i implies j* if and only if the following three conditions are satisfied:

1. $d_1(j) = d_1(i)$,
2. $d_2(j) = d_2(i)$,
3. $j < i$.

Observation 2 For any two distinct and intersecting edges i and j, either i implies j or j implies i or both. For any two disjoint edges, none of them implies the other.

Primary edges.

Definition 12 (Primary edges) Given a line assignment,

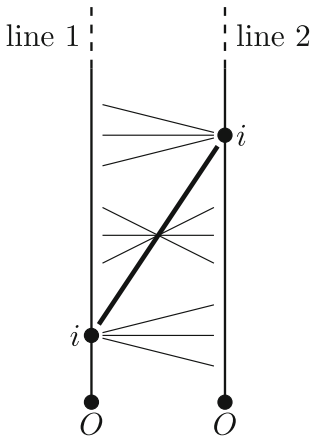


Fig. 4 Edge i implies all the other edges represented in the figure

- a non-horizontal edge is *primary* if and only if it is upward-oriented and it is not implied by any other upward-oriented edge;
- a horizontal edge is *primary* if and only if these three statements hold:
  1. it is 2-oriented,
  2. it is not implied by any upward-oriented edge,
  3. it is not implied by any 2-oriented horizontal edge.

Property 7 (Disjoint primary edges) *In any line assignment all primary edges are disjoint.*

Proof By contradiction, if two primary edges i and j intersect, then three cases may occur.

- Case 1: both edges are non-horizontal. Then by Observation 2 at least one of the edges implies the other one. Both edges are upward-oriented, because they are primary, thus contradicting Def. 12.
- Case 2: one of the two edges, say j, is horizontal and the other is not. By Def. 10 and Observation 2, i implies j. Edge i must be upward-oriented and edge j must be 2-oriented, because they are primary. This contradicts the assumption that j is primary.
- Case 3: both edges are horizontal (and coincident): then w.l.o.g. assume i implies j. Both edges must be 2-oriented, because they are primary. Then, by Def. 12, j cannot be primary, yielding a contradiction. □

Orientation of implied edges.
Now, by combining the definition of non-dominated line assignment with the definition of primary edges, we prove that the selection of the primary edges completely determines a non-dominated line assignment.

First, we state some properties on edge orientations implied by the assumption that an edge is primary in a non-dominated line assignment.

Property 8 (Non-horizontal primary edge) *If a non-horizontal edge $i \in N$ is primary in a non-dominated line assignment $\mathcal{A}$, then*

1. *edge i is upward-oriented;*
2. *each edge j implying i is downward-oriented;*
3. *each edge j implied by i is oriented to $L(i, \mathcal{A})$.*

Proof Statements 1 and 2 are implied by Def. 12, since i is primary. To prove statement 3, let ℓ be the line different from $L(i, \mathcal{A})$. Owing to Def. 10, if i implies j and i is upward-oriented to line $L(i, \mathcal{A})$, then $d_\ell(j) \geq d_\ell(i)$ and if the two edges coincide, then $j \prec i$. By contradiction, if j is oriented to ℓ, consider two cases. If the two edges do not coincide, then by Property 4, statement 1, the line assignment $\mathcal{A}$ is dominated, contradicting the assumption. If the two edges coincide, then by Property 4, statement 2, the line assignment $\mathcal{A}$ is dominated, contradicting the assumption. □

Property 9 (Horizontal primary edge) *If a horizontal edge $i \in N$ is primary in a non-dominated line assignment $\mathcal{A}$, then*

1. *edge i is 2-oriented;*
2. *all non-horizontal edges implying i are downward-oriented;*
3. *all horizontal edges implying i are 1-oriented;*
4. *all horizontal edges implied by i are 2-oriented.*

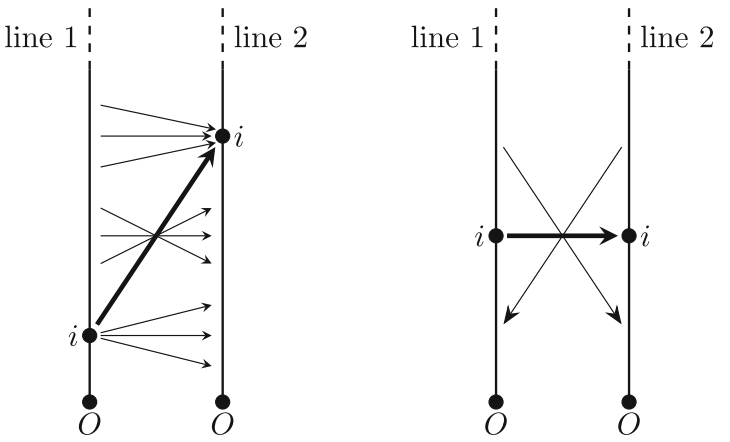
Proof Statements (1), (2) and (3) are implied by Def. 12. To prove statement (4), observe that, by Def. 11, an edge j implied by i must be horizontal, coincident with i and such that $j \prec i$. If j is 1-oriented, then edge reversal 3 can be applied to $\mathcal{A}$, showing that $\mathcal{A}$ is dominated. □

Properties 8 and 9 are illustrated in Figs. 5a and 5b. We now state some properties on edge orientations implied by the assumption that an edge is *not* primary in a non-dominated line assignment.

Definition 13 (Partial order) For each pair of distinct edges $i \in N$ and $j \in N$, i *precedes* j (indicated by $i \prec j$) if and only if $d_\ell(i) < d_\ell(j)\ \forall \ell = 1, 2$.

Definition 7 and Def. 13 directly give the following:

Observation 3 For each pair of disjoint edges $i \in N$ and $j \in N$, either $i \prec j$ or $j \prec i$. For each pair of intersecting edges $i \in N$ and $j \in N$, neither $i \prec j$ nor $j \prec i$.



(a) Non-horizontal primary edge. (b) Horizontal primary edge.

Fig. 5 The selection of edge i as a primary edge induces the orientation of edge i and all edges intersecting it

Property 10 (Non-primary edge) *Consider two edges $i \in N$ and $j \in N$ with $i \prec j$ that are consecutive primary edges in a non-dominated line assignment $\mathcal{A}$, i.e., there exists no primary edge $k \in N$ with $i \prec k \prec j$ in $\mathcal{A}$. Then,*

- *every non-horizontal edge $k \in N$ s.t. $i \prec k \prec j$ is downward-oriented;*
- *every horizontal edge $k \in N$ s.t. $i \prec k \prec j$ is* 1*-oriented.*

Proof The proof is by contradiction.

Case 1: Assume there exists a non-horizontal edge k between i and j that is upward-oriented in $\mathcal{A}$. Since i and j are consecutive primary edges, k cannot be primary. Therefore at least one of the conditions stated in Def. 12 must be violated by k. Since k is non-horizontal and upward-oriented, there must exist an upward-oriented edge e implying k. If e and k do not coincide and they are assigned to distinct lines, then the edge-reversal operation 1 applies, contradicting the assumption that $\mathcal{A}$ is non-dominated; then e and k are assigned to the same line (this trivially holds when the two edges coincide). Let ℓ' be the line $L(j, \mathcal{A}) = L(e, \mathcal{A})$ and ℓ'' the other line. Then $d_{\ell'}(e) \geq d_{\ell'}(k) > d_{\ell'}(i)$ and $d_{\ell''}(e) \leq d_{\ell''}(k) < d_{\ell''}(j)$. By Property 7, since i and j are primary, then edge e cannot intersect any of them. Therefore, e is an upward-oriented edge such that $i \prec e \prec j$; this would allow to repeat the same argument indefinitely, which is impossible since the number of edges between i and j is finite.

Case 2: Assume there exists a horizontal edge k between i and j that is 2-oriented in $\mathcal{A}$. Since i and j are consecutive primary edges, k cannot be primary. Therefore at least one of the conditions stated in Def. 12 must be violated. So, either k is implied by an upward-oriented edge or k is implied by a 2-oriented horizontal edge. In the former case, the proof for Case 1 applies. In the latter case, there exists another horizontal edge e coinciding with k and such that $e > k$, so that $i \prec e \prec j$; this would allow to repeat the same argument indefinitely, which is impossible since the number of edges between i and j is finite.

Because of the contradiction, all non-horizontal edges between i and j must be downward-oriented and all horizontal edges between i and j must be 1-oriented. □

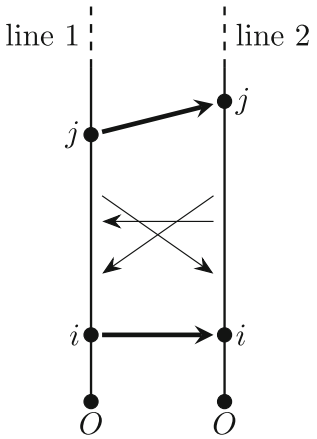


Fig. 6 The orientation of all edges between two consecutive primary edges i and j is determined by Property 10

Property 10 is illustrated by Fig. 6.

As a consequence of the properties above, if the search is restricted to non-dominated line assignments, once the primary edges have been selected the orientation of *all* the other edges follows.

Definition 14 (Primary set) The *primary set* of a solution is the set of its primary edges.

By Properties 8, 9 and 10, there exists a unique non-dominated line assignment $\mathcal{A}(\mathcal{P})$ having $\mathcal{P}$ as primary set.

3.3 Partial line assignments

In this subsection, we study the properties of partial line assignments, corresponding to partial primary sets. The goal is to design a dynamic programming algorithm that implicitly enumerates all primary sets by iteratively adding primary edges to partial primary sets in all possible ways. The iterations stop when all primary edges are selected and a line assignment (and its corresponding solution) is determined. Obviously, such a procedure would have complexity exponential in n. In order to manage a polynomially bounded number of partial primary sets, the idea is to extend the dominance properties between line assignments to dominance properties between partial line assignments. This allows to discard some of them earlier during the enumeration.

For this purpose, we exploit the partial order defined in Definition 13 and the properties of primary edges to prove that for each given partial primary set, it is possible to partition the edges into two subsets, such that the edge orientation in one of them is completely determined while the edge orientation in the other is completely free. From this intermediate step, we derive a dominance test to possibly discard partial line assignments.

Partial primary sets and edge partitions.
First of all, we prove that for any given partial primary set, the edges can be partitioned into two subsets, one with a fixed orientation and the other without any constraint on the orientation.

Definition 15 (Edge partition) For each edge $i \in N$, we define three subsets of items in which N is partitioned:

- $N^-(i) = \{j \in N : j \prec i\}$;
- $N^+(i) = \{j \in N : i \prec j\}$;
- $N^\pm(i) = N \backslash (N^-(i) \cup N^+(i))$.

Property 11 (Item positions) *For any non-dominated line assignment $\mathcal{A}$ in which edge $i \in N$ is a primary edge,*

1. $d_{L(j,\mathcal{A})}(j) < d_{L(j,\mathcal{A})}(i) \ \ \forall j \in N^-(i)$;
2. $d_{L(j,\mathcal{A})}(j) > d_{L(j,\mathcal{A})}(i) \ \ \forall j \in N^+(i)$;
3. $d_{L(j,\mathcal{A})}(j) \leq d_{L(j,\mathcal{A})}(i) \ \ \forall j \in N^\pm(i)$.

Proof Statements 1 and 2 directly follow from Definitions 13 and 15.

Statement 3 follows from Properties 8, 9 and Observation 2, stating that at least one of two intersecting edges must imply the other.

If edge i is non-horizontal and primary, then it is upward-oriented. If edge i implies $j \in N^\pm(i)$ and i is primary, then for Property 8 j is oriented to the same line as i, i.e., $L(j,\mathcal{A}) = L(i,\mathcal{A})$ and hence, by Def. 10, $d_{L(j,\mathcal{A})}(j) \leq d_{L(j,\mathcal{A})}(i)$.

If edge $j \in N^\pm(i)$ implies i and i is primary, then, for Property 8, j is downward-oriented and, by Def. 10, $d_{L(j,\mathcal{A})}(j) \leq d_{L(j,\mathcal{A})}(i)$.

If edge i is horizontal and $j \in N^\pm(i)$ is non-horizontal, then j implies i. Hence, by Property 9, if i is primary then j is downward-oriented and, by Def. 10, $d_{L(j,\mathcal{A})}(j) \leq d_{L(j,\mathcal{A})}(i)$.

If edge i is horizontal and $j \in N^\pm(i)$ is horizontal, then, by Def. 7, j coincides with i and therefore $d_{L(j,\mathcal{A})}(j) = d_{L(j,\mathcal{A})}(i)$. □

For any line assignment $\mathcal{A}$, let $N_\ell(\mathcal{A})$ the set of items assigned to line ℓ:

$$N_\ell(\mathcal{A}) = \{j \in N : L(j,\mathcal{A}) = \ell\} \ \ \forall \ell = 1, 2.$$

For any given primary item $i \in N$ of $\mathcal{A}$, consider the partition of $N_\ell(\mathcal{A})$ into two subsets:

$$S_\ell(i,\mathcal{A}) = \{j \in N_\ell(\mathcal{A}) : d_\ell(j) > d_\ell(i)\}$$

and its complement

$$R_\ell(i,\mathcal{A}) = \{j \in N_\ell(\mathcal{A}) : d_\ell(j) \leq d_\ell(i)\}.$$

Property 12 (Independent subsets) *If $\mathcal{A}$ is non-dominated,*

- *the elements in $S_\ell(i,\mathcal{A})$ are determined by the orientation of the edges in $N^+(i)$ and not by the orientation of the edges in $N^-(i) \cup N^\pm(i)$.*
- *The elements in $R_\ell(i,\mathcal{A})$ are determined by the orientation of the edges in $N^-(i) \cup N^\pm(i)$ and not by the orientation of the edges in $N^+(i)$.*

Proof The property immediately follows from Property 11 and the above definitions of $S_\ell(i,\mathcal{A})$ and $R_\ell(i,\mathcal{A})$. □

Property 12 is illustrated by Fig. 7.

Consider now two consecutive primary items i and j in a non-dominated line assignment $\mathcal{A}$, such that $i \prec j$. Let N_ℓ^{ij} be the set of edges in $(N^-(j) \cup N^\pm(j)) \cap N^+(i)$ that are oriented to line ℓ in $\mathcal{A}$, that is,

$$N_\ell^{ij} = S_\ell(i,\mathcal{A}) \cap R_\ell(j,\mathcal{A}) \ \ \forall i, j \in N \text{ primary edges s.t. } i \prec j.$$

We slightly extend this definition to include the edges before the first primary edge and after the last one. Let us introduce a dummy edge 0 preceding all edges in N and a dummy edge $n+1$ preceded by all edges in N. We define $N_\ell^{0j} = R_\ell(j,\mathcal{A}) \ \forall j = 1, \ldots, n$, $N_\ell^{i,n+1} = S_\ell(i,\mathcal{A}) \ \forall i = 1, \ldots, n$ and $N_\ell^{0,n+1} = N_\ell$. In this way, N_ℓ^{ij} is well-defined for all (i,j) pairs with $i = 0, \ldots, n$ and $j = 1, \ldots, n+1$ and $i \prec j$.

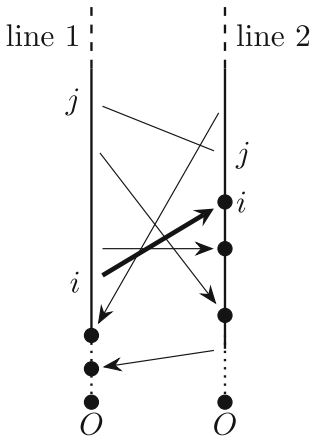


Fig. 7 When a partial primary set is defined up to edge i, the orientation of all edges in $R_\ell(i,\mathcal{A})$ is defined on each line, while the orientation of all edges in $S_\ell(i,\mathcal{A})$ is not constrained. All locations to be visited closer to O than the endpoints of edge i are determined (black dots), while all locations to be visited farther from O than the endpoints of edge i are undetermined (e.g., edge j)

Property 13 (Assignments between consecutive primary edges) *In any non-dominated line assignment $\mathcal{A}$ in which $i \in N \cup \{0\}$ and $j \in N \cup \{n+1\}$ with $i \prec j$ are consecutive primary edges, the elements in N_ℓ^{ij} are determined only by the primary edges i and j.*

Proof The set N_ℓ^{ij} is the intersection between $S_\ell(i,\mathcal{A})$ and $R_\ell(j,\mathcal{A})$. By Property 12, $S_\ell(i,\mathcal{A})$ only depends on the orientation of the edges in $N^+(i)$ and $R_\ell(j,\mathcal{A})$ only depends on the orientation of the edges in $N^-(j) \cup N^\pm(j)$. Since i and j are consecutive primary edges in $\mathcal{A}$, the only primary edges in $(N^-(j) \cup N^\pm(j)) \cap N^+(i)$ are edges i and j. Hence N_ℓ^{ij} only depends on primary edges i and j and not on the other primary edges in $\mathcal{A}$. □

Therefore, the elements of each set N_ℓ^{ij} can be computed by Properties 8, 9 and 10.

Cost of partial line assignments.

Not only the partial line assignment but also its cost can be derived from any given partial primary set. The cost of a set of trips on a line depends on the leading items of the trips. In turn, the leading items are determined by the greedy algorithm of Brucker et al. (1998) starting from the farthest items. Unfortunately, the construction of partial primary sets must proceed from the origin, i.e., starting from the closest items. For this reason it is not possible to determine the cost implied by the oriented edges in a partial primary set, because it is not known which items among them are leading in their trips. However, the number of possibilities is given by q. In other words, for any given partial primary set, the total cost corresponding to the leading items in the subset of oriented edges may have q distinct values. Therefore, a dynamic programming algorithm may associate q distinct states with each partial non-dominated primary set.

Given a non-dominated line assignment $\mathcal{A}$ and a primary edge $i \in N$, we define the number of *residual items* on each line at edge i as the value $r_\ell(i,\mathcal{A}) \in \{0,1,\ldots,q-1\}$ such that

$$r_\ell(i,\mathcal{A}) = |S_\ell(i,\mathcal{A})| \mod q \quad \forall \ell = 1,2.$$

Let $\mathcal{T}(\mathcal{A})$ be the compact solution obtained from line assignment $\mathcal{A}$ through the greedy algorithm of Brucker et al. (1998). Its cost is the sum of two contributions for each line ℓ: the cost of the trips having their leading item in $S_\ell(i,\mathcal{A})$ and the cost of the trips having their leading item in $R_\ell(i,\mathcal{A})$. We give formulas for these two contributions by assuming that $S_\ell(i,\mathcal{A})$ and $R_\ell(i,\mathcal{A})$ are represented as vectors indexed from 1 and whose entries are sorted by non-increasing distances from O on line ℓ. Denoting by $S_\ell(i,\mathcal{A})[t]$ and $R_\ell(i,\mathcal{A})[t]$ the t-th entry of such vectors, the sets of leading items in $S_\ell(i,\mathcal{A})$ and $R_\ell(i,\mathcal{A})$ are

$$\mathcal{L}_\ell^S(i,\mathcal{A}) = \{S_\ell(i,\mathcal{A})[t] : t \mod q = 1\}$$

and

$$\mathcal{L}_\ell^R(i,\mathcal{A}) = \{R_\ell(i,\mathcal{A})[t] : (t + r_\ell) \mod q = 1\}$$

respectively. Then the first cost contribution is

$$C_\ell(S_\ell(i,\mathcal{A})) = \sum_{k \in \mathcal{L}_\ell^S} d_\ell(k).$$

By Property 12, this sum includes the cost terms given by the edges in $N^+(i)$, and it does not depend on the orientation of the edges in $N^-(i) \cup N^\pm(i)$. Analogously, the second cost contribution is

$$C_\ell(R_\ell(i,\mathcal{A}), r_\ell) = \sum_{k \in \mathcal{L}_\ell^R} d_\ell(k). \tag{1}$$

By Property 12, this sum includes the cost terms given by the edges in $N^-(i) \cup N^\pm(i)$ and it does not depend on the orientation of the edges in $N^+(i)$, but only on the number of residual items $r_\ell(i,\mathcal{A})$ on each line. Setting $C^+(i,\mathcal{A}) = \sum_{\ell=1}^2 C_\ell(S_\ell(i,\mathcal{A}))$ and $C^-(i,\mathcal{A},r_1,r_2) = \sum_{\ell=1}^2 C_\ell(R_\ell(i,\mathcal{A}),r_\ell)$, we get that the cost $C(\mathcal{A})$ of solution $\mathcal{T}(\mathcal{A})$ is

$$C(\mathcal{A}) = C^+(i,\mathcal{A}) + C^-(i,\mathcal{A},r_1,r_2). \tag{2}$$

Partial line assignments and dominance.

As a consequence of the above properties, an optimal solution of $q/1/P/V$ can be found by (implicitly) enumerating all primary sets. This is done by the algorithm illustrated in Sect. 4.

The algorithm iteratively defines primary sets by adding a primary item at each iteration. At intermediate iterations, this generates partial line assignments, as defined below.

Definition 16 (Partial line assignments) For any given edge $i \in N$, a *partial line assignment* $\mathcal{A}_i$ is an assignment to the lines of all items in $N^-(i) \cup N^{\pm}(i)$ so that i is primary. The corresponding *partial primary set* $\mathcal{P}_i$ is the set of primary items of $\mathcal{A}_i$. A (possibly partial) line assignment $\mathcal{A}$ *extends* $\mathcal{A}_i$ if and only if the items in $N^-(i) \cup N^{\pm}(i)$ are assigned to the same lines in both $\mathcal{A}$ and $\mathcal{A}_i$.

Property 14 (Dominance between partial line assignments) *For a given $i \in N$, consider two partial line assignments $\mathcal{A}'_i$ and $\mathcal{A}''_i$ (hence i is primary for both) and a line assignment $\mathcal{A}''$ extending $\mathcal{A}''_i$. Let also $r_\ell = r_\ell(i, \mathcal{A}'')$ for $\ell = 1, 2$. Then, if $C^-(i, \mathcal{A}'_i, r_1, r_2) < C^-(i, \mathcal{A}''_i, r_1, r_2)$ (resp. $C^-(i, \mathcal{A}'_i, r_1, r_2) = C^-(i, \mathcal{A}''_i, r_1, r_2)$), there exists $\mathcal{A}'$ extending $\mathcal{A}'_i$ such that $C(\mathcal{A}') < C(\mathcal{A}'')$ (resp. $C(\mathcal{A}') = C(\mathcal{A}'')$).*

Proof Given $\mathcal{A}''$ extending $\mathcal{A}''_i$, define the line assignment $\mathcal{A}'$ from $\mathcal{A}''$ by orienting the edges in $N^-(i) \cup N^{\pm}(i)$ as in $\mathcal{A}'_i$. Then, by Def. 16, $\mathcal{A}'$ extends $\mathcal{A}'_i$. By Property 11 $S_\ell(i, \mathcal{A}') = S_\ell(i, \mathcal{A}'')$ since such sets only depend on the orientation of the edges in $N^+(i)$ which is the same in $\mathcal{A}'$ and $\mathcal{A}''$ by construction. Hence, $|S_\ell(i, \mathcal{A}'')| \mod q = r_\ell \ \forall \ell = 1, 2$ implies $|S_\ell(i, \mathcal{A}')| \mod q = r_\ell \ \forall \ell = 1, 2$. Moreover, the same argument yields $\mathcal{L}^S_\ell(i, \mathcal{A}'') = \mathcal{L}^S_\ell(i, \mathcal{A}')$ for $\ell = 1, 2$, thus $C^+(i, \mathcal{A}'') = C^+(i, \mathcal{A}')$. Then the result follows from Eqn. (2). □

Property 14 allows to define a dominance relation between partial line assignments, analogous to the dominance relation between line assignments stated in Property 4. Hence, dominated partial line assignments can be immediately discarded without losing the guarantee that an optimal solution will be eventually found.

This can be seen as a special case of Bellman optimality principle and it is the basis for a dynamic programming algorithm: a sub-policy characterized by $\mathcal{P}'_i$ dominates a sub-policy characterized by $\mathcal{P}''_i$ for a given pair of residual values (r_1, r_2) if and only if $C^-(i, \mathcal{A}'_i, r_1, r_2) < C^-(i, \mathcal{A}''_i, r_1, r_2)$. In case of tie, any arbitrary criterion can be used to discard one of the two equivalent partial solutions.

4 A dynamic programming algorithm

In this section, we provide a description of a dynamic programming algorithm that solves $q/1/P/V$ to optimality, relying on the properties illustrated in the previous sections.

States.

A state of the dynamic programming algorithm is a triple $\{i, r_1, r_2\}$, with $i \in N \cup \{0, n+1\}$ and r_1 and r_2 satisfying

$$(r_1 + r_2) \mod q = \rho_i,$$

where $\rho_i = |N^+(i)| \mod q$. Hence, there are q distinct states for each item $i \in N \cup \{0\}$ and the following relations allow to obtain r_1 from r_2 and vice versa:

$$r_1 = (\rho_i - r_2) \mod q$$
$$r_2 = (\rho_i - r_1) \mod q.$$

A state $\{i, r_1, r_2\}$ with $i \in N$ corresponds to a partial line assignment $\mathcal{A}_i$ having i as primary item. Triples of the form $\{0, r_1, r_2\}$ are called *initial states*: in this case r_1 and r_2 indicate the number of residual items on each line before orienting any edge. These q initial states are given by the q possible values of (r_1, r_2) pairs such that $(r_1 + r_2) \mod q = n \mod q$, that is, $\{(0, \rho), \ldots, (\rho, 0), (\rho + 1, q - 1), \ldots, (q - 1, \rho + 1)\}$, where $\rho = n \mod q$.

The *final state* $\{n + 1, 0, 0\}$ corresponds to a full line assignment without residual items; hence, it must be reached to guarantee that on each line the leading item of the farthest trip is a farthest item.

Extension rule.

States are iteratively extended according to a pre-topological order of the items, with the addition of the initial item 0 preceding all items in N and the final item $n+1$ following all items in N. A set W of edges is initialized at N. Then, at each iteration the states of an edge $j \in W$ are evaluated and then j is deleted from W. The algorithm terminates when $W = \emptyset$, meaning that the costs of all states have been computed. The selection of $j \in W$ obeys the rule that all predecessor states must have already been evaluated, i.e., none of them must be in W. Initially, only the initial states satisfy this condition.

Extending a state from a predecessor state $\{i, r_1, r_2\}$ to a successor state $\{j, r'_1, r'_2\}$ means extending the line assignment $\mathcal{A}_i$ corresponding $\{i, r_1, r_2\}$ with the line assignment $\mathcal{A}_j$ such that edge j is primary in $\mathcal{A}_j$ and no primary edge exists between the primary edges i and j.

Owing to Property 13 and the definition of residual items, $r_\ell = (r'_\ell + |N^{ij}_\ell|) \mod q \ \ \forall \ell \in \{1, 2\}$. In this way a one-to-one correspondence is established between the q states of edge i and the q states of edge j for each (i, j) pair such that $i \prec j$. For each state $\{j, r'_1, r'_2\}$, we indicate by $Pred(j, r'_1, r'_2)$ the set of its predecessor states:

$$Pred(j, r'_1, r'_2) = \{\{i, r_1, r_2\} : i \prec j \text{ and } r_\ell = (r'_\ell + |N^{ij}_\ell|) \mod q \ \ \forall \ell \in \{1, 2\}\}.$$

We remark that $\rho_i = (\rho_j + |N_1^{ij}| + |N_2^{ij}|) \mod q$. We also remark that for the predecessor $\{i, r_1, r_2\}$ of $\{j, r_1', r_2'\}$ it holds $r_\ell' = (r_\ell - |N_\ell^{ij}|) \mod q \ \ \forall \ell = 1, 2$.

Cost extension.

Each state $\{i, r_1, r_2\}$ has an associated cost $C(i, r_1, r_2)$, that is the minimum cost of a partial solution corresponding to the state, owing to Property 14.

The cost associated with the initial states is null: $C(0, r_1, r_2) = 0 \ \forall (r_1, r_2) : (r_1 + r_2) \mod q = n \mod q$. The cost associated with a successor state $\{j, r_1', r_2'\}$ is computed from that of a predecessor state $\{i, r_1, r_2\}$ through a cost extension function $\Delta(i, j, r_1', r_2')$ given below. Only the minimum cost is retained, according to the dominance criterion established in Property 14:

$$C(j, r_1', r_2') = \min_{\{i, r_1, r_2\} \in Pred(j, r_1', r_2')} \{C(i, r_1, r_2) + \Delta(i, j, r_1', r_2')\}.$$

Consistently with Eqn. (1), $\Delta(i, j, r_1', r_2')$ is the sum of the distances of the leading items in N_1^{ij} and N_2^{ij}. These can be identified according to the values of r_1' and r_2'. For each line $\ell \in \{1, 2\}$, consider an array made of the edges in N_ℓ^{ij} indexed from 1 and sorted by non-increasing value of distance from O. Let $N_\ell^{ij}[t]$ be the edge in position t in the array. Then, the set of leading items in $N_\ell^{ij}[t]$ is

$$\mathcal{L}_\ell^{ij} = \{N_\ell^{ij}[t] : (t + r_\ell') \mod q = 1\}.$$

Then,

$$\Delta(i, j, r_1', r_2') = \sum_{\ell=1}^{2} \sum_{k \in \mathcal{L}_\ell^{ij}} d_\ell(k).$$

For the sake of completeness and rigorousness of presentation, the complete pseudo-code of the dynamic programming algorithm is reported in the Appendix.

Complexity.

There are $O(nq)$ states. The number of (i, j) pairs such that $i \prec j$ is $O(n^2)$; q values of $\Delta(i, j)$ must be computed for each pair. Overall, by using a suitable procedure on N_ℓ^{ij}, such q values can be computed in $O(n)$ time. Therefore the asymptotic worst-case time complexity of the algorithm is $O(n^3)$.

5 Conclusions and further extensions

Proving that problem $q/1/P/V$ is polynomially solvable is a significant step forward in the analysis and classification of the many variations of single-crane picking problem in AS/RS. The complexity of this variation had been left open in a previous work (see Barbato et al. (2019)), where several polynomial-time variations had been identified.

Another open problem variation is $2/1/PD/F$, that is the problem on a single line, where the origin is at an endpoint, the crane capacity is equal to 2, locations are unique for each item and both pickup orders (retrieval operations, in any sequence) and delivery orders (storage operations, according to a given sequence) must be executed.

The identification of polynomially solvable variations, although they may look oversimplified compared to real problems, is important to provide efficiently solvable relaxations that can be exploited within branch-and-bound algorithms to solve more complex (NP-hard) and realistic variations.

A possibly interesting extension of this work on the $q/1/P/V$ variation concerns the problem with $L \geq 2$ aisles, that can be decomposed into L single-aisle sub-problems, to be efficiently solved with the algorithm presented here. Also, problems in 2 dimensions can be solved via exact optimization algorithms, such as branch-and-bound, where a polynomial-time algorithm for the 1-dimensional problem is useful to efficiently compute lower bounds.

Appendix A Pseudo-code

The procedure to solve problem $q/1/P/V$ can be split in three subroutines, as indicated in the pseudo-code of Algorithm 1.

Algorithm 1 Main algorithm

```
1: Preprocessing
2: OrientEdges
3: DynamicProgramming
```

The Preprocessing step determines the subsets N^-, N^+ and $N^\pm$ of Definition 15. Its pseudo-code, provided in Algorithm 2, is a simple `if-else` procedure that considers all possible positions of pairs of distinct items on both lines.

Procedure OrientEdges, described in Algorithm 3, considers all potential pairs of consecutive primary items $i, j \in N$ (checking that $i \prec j$, line 4). For each such pair, it orients the edges with at least one endpoint between i and j on any line in the (unique) way guaranteeing that the resulting partial orientation belongs to a non-dominated solution having i and j as consecutive primary items. This is done by imposing separately the conditions of Property 10 (lines 5–11), Property 9 (lines 13–28) and Property 8 (lines 30–42). Each block imposing one of these conditions is essentially an exhaustive enumeration of all possible cases and takes $O(n)$ time. So, overall, the whole OrientEdges procedures takes $O(n^3)$ time. Note also that, for $\ell = 1, 2$ and for every i, j

Algorithm 2 Pre-processing

```
1: procedure Preprocessing(N, d) Output: N^-(e), N^+(e) and N^±(e) for e ∈ N
2:     for i = 1, ..., n do
3:         N^+(i) ← ∅
4:         N^-(i) ← ∅
5:         N^±(i) ← {i}
6:     end for
7:     for i = 1, ..., n − 1 do
8:         for j = i + 1, ..., n do
9:             if (d1(i) < d1(j)) ∧ (d2(i) < d2(j)) then
10:                N^+(i) ← N^+(i) ∪ {j}
11:                N^-(j) ← N^-(j) ∪ {i}
12:            else
13:                if (d1(i) > d1(j)) ∧ (d2(i) > d2(j)) then
14:                    N^+(j) ← N^+(j) ∪ {i}
15:                    N^-(i) ← N^-(i) ∪ {j}
16:                else
17:                    N^±(j) ← N^±(j) ∪ {i}
18:                    N^±(i) ← N^±(i) ∪ {j}
19:                end if
20:            end if
21:        end for
22:    end for
23: end procedure
```

satisfying the **if** condition of line 4, the set N_ℓ^{ij} has $O(n)$ elements.

The third step DynamicProgramming, described in Algorithm 4, is the complete dynamic programming procedure solving problem $q/1/P/V$. It has three main blocks: initialization, cost extension and extensions to the final state.

The initialization (lines 2–6) defines all possible initial states and sets their cost to 0.

Next, the set W of items to be processed is defined. Here "processing" an item means to compute the costs of all its possible states starting from the costs of its predecessor states. Initially W contains all items. Then, sequentially a new item is processed and, after its processing, it is removed from W (line 38). The item selection on line 9 ensures that the states of the next item to be processed have all their predecessor states already processed (see paragraph Cost Extension of Sect. 4 for the definition of predecessor state). Given the current item j to be processed, lines 15 and 16 select the items $i \in N$ having a predecessor state of some state of j. The variable Δ used in Algorithm 4 can be interpreted as a 2-dimensional matrix for each fixed $i, j \in N$. Its entry (r_1', r_2') contains the value by which the cost $C(j, r_1', r_2')$ of state (j, r_1', r_2') is computed. All entries of Δ are initially set to 0 (lines 17–20); subsequently their actual values are computed

Algorithm 3 Orienting the edges between consecutive primary edges

```
1: procedure OrientEdges(N, d, N^+(e), N^-(e), N^±(e) for every e ∈ N) Output: N_ℓ^ij for i, j ∈ N and ℓ = 1, 2
2:     for i = 0, ..., n do
3:         for j = 1, ..., n + 1 do
4:             if i ≺ j then
               // Edges disjoint from i and j: apply Property 10
5:                 for k ∈ N^+(i) ∩ N^-(j) do
6:                     if d1(k) ≤ d2(k) then
7:                         N_1^ij ← N_1^ij ∪ {k}
8:                     else
9:                         N_2^ij ← N_2^ij ∪ {k}
10:                    end if
11:                end for
                   // Edges intersecting j
12:                if d1(j) = d2(j) then
                   // j is horizontal: apply Property 9
13:                    N_2^ij ← N_2^ij ∪ {j}
14:                    for k ∈ N^±(j) do
15:                        if d1(k) ≠ d2(k) then
16:                            if d1(k) < d2(k) then
17:                                N_1^ij ← N_1^ij ∪ {k}
18:                            else
19:                                N_2^ij ← N_2^ij ∪ {k}
20:                            end if
21:                        else
22:                            if k > j then
23:                                N_1^ij ← N_1^ij ∪ {k}
24:                            else
25:                                N_2^ij ← N_2^ij ∪ {k}
26:                            end if
27:                        end if
28:                    end for
29:                else
                   // j is not horizontal: apply Property 8
30:                    if d1(j) < d2(j) then
31:                        λ ← 2
32:                    else
33:                        λ ← 1
34:                    end if
35:                    N_λ^ij ← N_λ^ij ∪ {j}
36:                    for k ∈ N^±(j) do
37:                        if (d_λ(k) > d_λ(j)) ∨ (d_{3−λ}(k) < d_{3−λ}(j)) ∨ ((d_λ(k) = d_λ(j)) ∧ (d_{3−λ}(k) = d_{3−λ}(j)) ∧ (k > j)) then
38:                            N_{3−λ}^ij ← N_{3−λ}^ij ∪ {k}
39:                        else
40:                            N_λ^ij ← N_λ^ij ∪ {k}
41:                        end if
42:                    end for
43:                end if
44:            end if
45:        end for
46:    end for
47: end procedure
```

as follows (lines 43–48): for each item in $e \in N_{ij}$ (which is available from procedure OrientEdges) find the two states (i, r_1, r_2) and (j, r_1', r_2') for which e is a leading item and sum its distance from the origin to the current value of entry (r_1', r_2') of Δ. Note that, since $|N_{ij}| \in O(n)$, computing Δ for a given pair of items i, j takes $O(n)$ time. Finally, still for fixed i and j, the cost $C(j, r_1', r_2')$ of all states (j, r_1', r_2') associated to j set equal to the sum of $C(i, r_1, r_2)$ and of entry (r_1', r_2') of Δ whenever the resulting value improves over the current value of $C(j, r_1', r_2')$ (initially such value is $+\infty$, see lines 10–13). The computation of all q costs of a given j is performed in lines 30–35 in $O(nq)$ time.

Finally, the extension to the final state (lines 40–52) is similar to the cost extension subroutine just described but, instead of an actual item j, it is performed on the fictitious item $n + 1$.

The bottleneck of the whole DynamicProgramming step is the computation of Δ for each suitable pair of items i, j. Since Δ is computed in $O(n)$ time for fixed i, j, and there are $O(n^2)$ pairs, the algorithm runs in $O(n^3)$ time.

Algorithm 4 Dynamic programming

procedure DynamicProgramming
// Initialization
$\rho_0 \leftarrow n \mod q$
for $r_1 = 0, \ldots, q-1$ **do**
$r_2 \leftarrow (\rho_0 - r_1) \mod q$
$C[0, r_1, r_2] \leftarrow 0$
end for
$W \leftarrow N$
// Cost extension
while $W \neq \emptyset$ **do**
Select $j : N^-(j) \cap W = \emptyset$
// Initialize $C(j, \cdot, \cdot)$
for $r'_1 = 0, \ldots, q-1$ **do**
$r'_2 \leftarrow (\rho_j - r'_1) \mod q$
$C(j, r'_1, r'_2) \leftarrow +\infty$
end for
$\rho_j \leftarrow (|N^+(j)|) \mod q$
for $i \in N \cup \{0\}$ **do**
if $i \prec j$ **then**
// Initialize Δ
for $r'_1 = 0, \ldots, q-1$ **do**
$r'_2 \leftarrow (\rho_j - r'_1) \mod q$
$\Delta[i, j, r'_1, r'_2] \leftarrow 0$
end for
// Compute the q values of $\Delta[i, j]$
for $l \in \{1, 2\}$ **do**
for $t = 1, \ldots, |N^{ij}_\ell|$ **do**
// Find the state for which component t corresponds to a leading item
$r'_\ell \leftarrow (1 - t) \mod q$
$r'_{3-\ell} \leftarrow (\rho_j - r'_\ell) \mod q$
$r_\ell \leftarrow (r'_\ell + |N^{ij}_\ell|) \mod q$
$r_{3-\ell} \leftarrow (\rho_i - r_\ell) \mod q$
$\Delta[i, j, r'_1, r'_2] \leftarrow \Delta[i, j, r'_1, r'_2] + d_\ell(N^{ij}_\ell[t])$
end for
end for
// Compare and retain the policy of minimum cost for each state of item j
for $r'_1 = 0, \ldots, q-1$ **do**
$r'_2 \leftarrow (\rho_j - r'_1) \mod q$
$r_1 \leftarrow (r'_1 - |N^{ij}_1|) \mod q$
$r_2 \leftarrow (r'_2 - |N^{ij}_2|) \mod q$
$C(j, r'_1, r'_2) \leftarrow \min\{C(j, r'_1, r'_2), C(i, r_1, r_2) + \Delta[i, j, r'_1, r'_2]\}$
end for
end if
end for
$W \leftarrow W \backslash \{j\}$
end while
// Extensions to the final state $n + 1$
$z \leftarrow \infty$
for $i = 0, \ldots, n$ **do**
// Initialize Δ
$\Delta[i, n+1] \leftarrow 0$
// Compute $\Delta[i, n+1]$
for $l \in \{1, 2\}$ **do**
for $t = 1, \ldots, |N^{i,n+1}_\ell|$ **do**
if $t \mod q = 1$ **then**
$\Delta[i, n+1] \leftarrow \Delta[i, n+1] + d_\ell(N^{i,n+1}_\ell[t])$
end if
end for
// Find the predecessor state
$r_\ell \leftarrow |N^{i,n+1}_\ell| \mod q$
end for
// Compare and retain the policy of minimum cost for the final state
$z \leftarrow \min\{z, C(i, r_1, r_2) + \Delta[i, n+1]\}$
end for
end procedure

Acknowledgements The authors are grateful to the two anonymous reviewers for their suggestions that substantially improved the paper. This research was partially funded by Regione Lombardia, grant agreement n. E97F17000000009, project AD-COM.

Funding Open access funding provided by Università degli Studi di Milano within the CRUI-CARE Agreement.

References

Azadeh, K., De Koster, R., & Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science, 53*(4), 917–945. https://doi.org/10.1287/trsc.2018.0873

Barbato, M., Ceselli, A., & Righini, G. (2019). Paths and matchings in an automated warehouse. In: M. Paolucci, A. Sciomachen, & P. Uberti (Eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises* (pp. 151–159). AIRO Springer Series, vol 3. Springer.

Boysen, N., de Koster, R., & Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research, 277*(2), 396–411. https://doi.org/10.1016/j.ejor.2018.08.023

Boysen, N., & Stephan, K. (2016). A survey on single crane scheduling in automated storage/retrieval systems. *European Journal of Operational Research, 254*(3), 691–704. https://doi.org/10.1016/j.ejor.2016.04.008

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M., Potts, C., Tautenhahn, T., & van de Velde, S. (1998). Scheduling a batching machine. *Journal of Scheduling, 1*(1), 31–54.

de Koster, R., Le-Duc, T., & Roodbergen, K. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research, 182*, 481–501. https://doi.org/10.1016/j.ejor.2006.07.009

Dooly, D. R., & Lee, H. F. (2008). A shift-based sequencing method for twin-shuttle automated storage and retrieval systems. *IIE Transactions, 40*(6), 586–594. https://doi.org/10.1080/07408170701730776

Fontin, J. R., & Lin, S. W. (2020). A joint comparative analysis of routing heuristics and paperless picking technologies using simulation and data envelopment analysis. *Applied Sciences, 10*(24), 8777.

Gharehgozli, A. H., Yu, Y., Zhang, X., & Koster, Rd. (2017). Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system. *Transportation Science, 51*(1), 19–33. https://doi.org/10.1287/trsc.2014.0562

Goeke, D., & Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing, 33*(2), 436–451. https://doi.org/10.1287/ijoc.2020.1040

Gu, J., Goetschalckx, M., & McGinnis, L. F. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research, 177*(1), 1–21. https://doi.org/10.1016/j.ejor.2006.02.025

Ouzidan, A., Sevaux, M., Olteanu, A., Pardo, E., & Duarte, A. (2022). On solving the order processing in picking workstations. *Optimization Letters, 16*, 5–35. https://doi.org/10.1007/s11590-020-01640-w

Roodbergen, K., & Vis, I. (2009). A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research, 194*(2), 343–362. https://doi.org/10.1016/j.ejor.2008.01.038

Tang, L., & Ren, H. (2010). Modelling and a segmented dynamic programming-based heuristic approach for the slab stack shuffling problem. *Computers and Operations Research, 37*, 368–375. https://doi.org/10.1016/j.cor.2009.05.011

van den Berg, J., & Zijm, W. (1999). Models for warehouse management: Classification and examples. *International Journal on Production Economics, 59*, 519–528. https://doi.org/10.1016/S0925-5273(98)00114-5

Weidinger, F., Boysen, N., & Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research, 274*, 501–515. https://doi.org/10.1016/j.ejor.2018.10.021