Boysen, Nils; Emde, Simon; Stephan, Konrad

Article

Crane scheduling for end-of-aisle picking: Complexity and efficient solutions based on the vehicle routing problem

EURO Journal on Transportation and Logistics (EJTL)

Provided in Cooperation with:
Association of European Operational Research Societies (EURO), Fribourg

Suggested Citation: Boysen, Nils; Emde, Simon; Stephan, Konrad (2022) : Crane scheduling for end-of-aisle picking: Complexity and efficient solutions based on the vehicle routing problem, EURO Journal on Transportation and Logistics (EJTL), ISSN 2192-4384, Elsevier, Amsterdam, Vol. 11, Iss. 1, pp. 1-11,
https://doi.org/10.1016/j.ejtl.2022.100085

This Version is available at:
https://hdl.handle.net/10419/325158

EURO Journal on Transportation and Logistics 11 (2022) 100085

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

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

Crane scheduling for end-of-aisle picking: Complexity and efficient solutions based on the vehicle routing problem

Nils Boysen [a,*], Simon Emde [b], Konrad Stephan [a]

[a] Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3, 07743 Jena, Germany
[b] CORAL - Cluster for Operations Research, Analytics, and Logistics, Department of Economics and Business Economics, Aarhus University, Fuglesangs Allé 4, Aarhus V DK-8210, Denmark

ARTICLE INFO

Keywords:
Facility logistics
Warehousing
Crane scheduling
Vehicle routing

ABSTRACT

To relieve human order pickers from unproductive walking through a warehouse, parts-to-picker systems deliver demanded stock keeping units (SKUs) toward picking workstations. In a wide-spread parts-to-picker setup, a crane-operated automated storage and retrieval system (ASRS) delivers bins with demanded SKUs toward an end-of-aisle picking workstation and returns them back into the rack once the picks are completed. We consider the scheduling of the crane that operates subsequent dual commands. Each dual command combines a retrieval request for another SKU bin demanded at the picking workstation with a storage request, where a bin that has already been processed and passed through the bin buffer is returned to its dedicated storage position in the ASRS. This system setup in general and the resulting crane scheduling problem in particular have been an active field of research for more than 30 years. We add the following contributions to this stream of research: We finally prove that the crane scheduling problem is strongly NP-hard. Furthermore, we show that, although only a single vehicle (namely, the crane) is applied, the problem is equivalent to the traditional vehicle routing problem (VRP). This opens the rich arsenal of very efficient VRP solvers, which substantially outperform existing tailor-made algorithms from the literature.

1. Introduction

In the wake of ever increasing e-commerce sales (e.g., see Statista, 2019), up-to-date warehouses more and more evolve into complex, technology-enriched, and mission-critical fulfillment factories. Traditional picker-to-parts warehouses are often not efficient enough to fulfill the ambitious service requirements of modern retailing. In these traditional systems, human pickers walk (or drive on a picking cart) from shelf to shelf when retrieving demanded stock keeping units (SKUs). To avoid the resulting unproductive walking (or driving) effort during order picking, which is frequently cited to account for 55% of the total warehouse operating expense (e.g. De Koster et al., 2007), parts-to-picker systems automate parts of the fulfillment process and trade off wage costs with long-term invest costs for material handling machinery.

Any parts-to-picker warehousing system consists of (at least) three parts. The *SKU supply subsystem* stores SKUs in bins (or other containers) and delivers them toward stationary pickers upon request. After picking, the bins are returned into the SKU supply system. Since retrieving pieces of SKUs with varying dimensions, stability, and uneven surface structure from bins is still hardly automatable these days, at least not with a competitive performance (Azadeh et al., 2019), human order pickers located in picking workstations constitute the *picking subsystem* of a parts-to-picker warehousing system. Furthermore, we have a conveyor system in between both previous subsystems that also serves as a *bin buffer* to uncouple the previous two subsystems.

We consider a basic and well-established parts-to-picker setup, where a crane-operated rack system supplies an end-of-aisle picking workstation. This basic setup is schematically depicted in Fig. 1. There exist alternative ASRSs, such as compact Autostore systems (Zou et al., 2018), carousels (Litvak and Vlasiou, 2010), and lift-and-shuttle systems (Azadeh et al., 2019). We, however, address the most widespread setup, with a tradition dating back to the 1950s (Allied Market Research, 2020). In a crane-operated racking system (see Fig. 1), a crane (also denoted as storage and retrieval machine), which moves on rails mounted to the floor and ceiling, services an aisle of racks to the left and right (Boysen and Stephan, 2016). For retrieving a SKU bin (or pallet or other container) from a shelf and returning it, the crane is equipped with a shuttle having some fork or conveyor mechanism.

* Corresponding author.
E-mail addresses: nils.boysen@uni-jena.de (N. Boysen), siem@econ.au.dk (S. Emde), konrad.stephan@uni-jena.de (K. Stephan).
URLs: http://www.om.uni-jena.de (N. Boysen), http://person.au.dk/en/ (S. Emde).

https://doi.org/10.1016/j.ejtl.2022.100085
Received 10 November 2021; Received in revised form 4 May 2022; Accepted 13 June 2022
Available online 14 June 2022

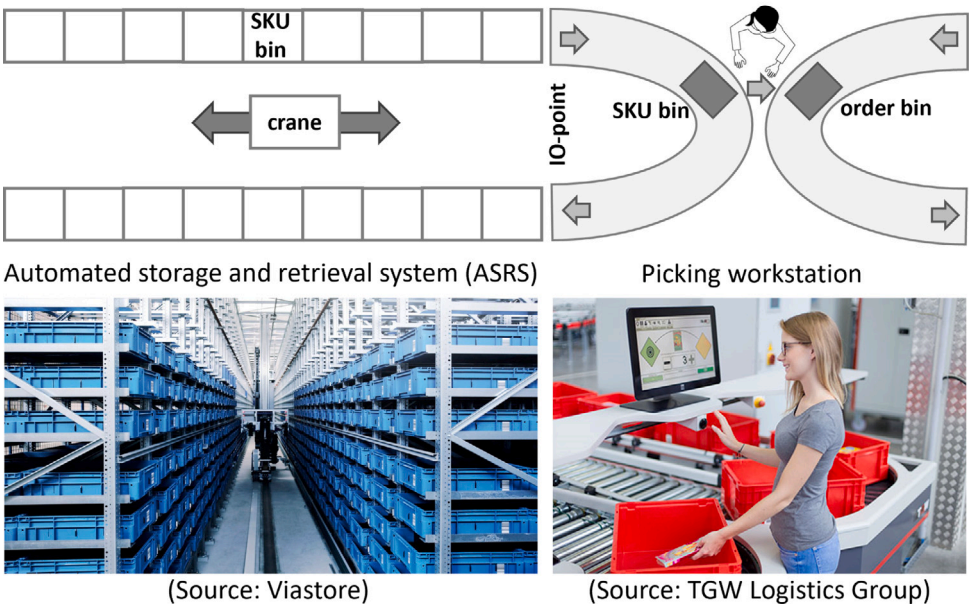

Fig. 1. Basic setup of an end-of-aisle picking workstation supplied by a crane-operated ASRS.

Once retrieved, the crane moves the current bin to the input output (IO) point, located at the front-end of the aisle. Here, the bin is placed onto a horseshoe conveyor that connects the picking subsystem. To reduce unproductive crane movement, so-called dual commands are the preferred mode of crane operations. Instead of moving back empty in order to fetch the next SKU bin, the crane loads another bin to be stored back into the ASRS when returning from the IO point. Thus, a dual command combines a storage and retrieval request between any two subsequent visits at the IO point (Boysen and Stephan, 2016).

The conveyor transports a SKU bin toward the end-of-aisle picking workstation, which is also denoted as the picking bay (Dallari et al., 2009). The conveyor also operates as a bin buffer that uncouples crane and picking workstation, so that neither of them has to wait for any delay of the other subsystem. Within the bin buffer, SKU bins successively move forward slot by slot until a SKU bin is activated and loaded into the picking workstation. In parallel to the progression of SKU bins, another kind of bin is loaded into the workstation. Such a bin is initially empty and associated with a specific picking order. It receives the demanded pieces of SKUs defined on its order's pick list, and we call this second kind of bins the *order bins*. Announced on a monitor or indicated by a put-to-light display, the picker is informed on the number of pieces of the current SKU demanded by the active order bin. She receives the pieces from the SKU bin, puts them into the respective order bin, and confirms the pick by pressing a button. Once an order bin has received all demanded pieces, it progresses onward to the shipping area and is replaced by the next order bin. SKU bins that are no longer demanded also move onward and are added to the queue of storage requests in front of the IO point, where they are subsequently returned in a first-in-first-out (FIFO) manner by the crane into the ASRS. In such a picking system, human order pickers can process up to 1000 order lines per hour (De Koster et al., 2007; De Koster, 2007), so that picking workstations are among the most efficient picking devices (Füßler and Boysen, 2019).

Given this widespread parts-to-picker setup, this paper aims to optimize the dual commands operated by the crane of the ASRS when supplying the end-of-aisle picking workstation with demanded SKU bins and returning them after the picks are completed. A SKU bin that is retrieved from the ASRS and delivered toward the picking workstation has to be stored back into the ASRS after passing the fixed-length bin buffer of the picking workstation, so that storage and retrieval requests are interdependent and cannot be combined without restriction. By minimizing the empty crane travel while processing the dual commands, the bin throughput of the ASRS, which due to its high investment costs is often the bottleneck resource (Füßler and Boysen, 2019), is increased. The end-of-aisle part-to-picker setup in general (Bozer and White, 1990; Foley et al., 2004) and the resulting crane scheduling problem in particular (Bozer and White, 1996; Mahajan et al., 1998; Baardman and Roodbergen, 2016; Baardman et al., 2021) have been under research for more than 30 years. In this paper, we add the following research contributions to this stream of research:

- We finally settle the computational complexity of the crane scheduling problem and prove that the problem is strongly NP-hard. Note that Baardman et al. (2021) show that instances of the end-of-aisle crane scheduling problem can be transformed into instances of a special case of the multiple traveling salesman problem (mTSP). However, this merely demonstrates that the mTSP is at least as hard as crane scheduling, which is unsurprising, given the mTSP's NP-hard nature, but does not prove NP-hardness of our crane scheduling problem. A more detailed explanation of this matter is given in the Appendix.
- In spite of this (on first sight daunting) result, we furthermore show that the problem, although it applies only a single vehicle (namely the crane), is a special case of the capacitated vehicle routing problem (VRP). This transformation opens up the rich arsenal of very efficient exact and heuristic VRP solvers developed in decades of research by the vehicle routing community. By applying a state-of-the-art VRP solver, we obtain high-quality solutions to large instances a matter of seconds or minutes, vastly outperforming previous heuristics from the literature in terms of both speed and solution quality.
- Finally, solving much larger instances allows us to systematically investigate the impact of differently sized bin buffers. Hence, we can derive some decision support for warehouse managers having to design an end-of-aisle picking system.

The remainder of the paper is structured as follows. Section 2 reviews the related literature. Then, 3 defines our crane scheduling problem. Computational complexity is investigated in Section 4, and equivalence with the well known vehicle routing problem is shown in Section 5. In Section 6, we benchmark our solution approach with those of previous research. Finally, Section 7 concludes the paper.

2. Literature review

We cannot provide an exhaustive survey on the research efforts related to the SKU supply and the picking subsystems, because either of them have attracted a considerable amount of papers. Instead, we refer to existing survey papers on crane operated ASRS by Gagliardi et al. (2012) as well as Roodbergen and Vis (2009), crane scheduling in ASRS by Boysen and Stephan (2016), and picking workstations by Boysen et al. (2019). In the following, we briefly characterize the most important decision problems when setting up and operating an end-of-aisle parts-to-picker system, survey important papers, and relate them to our specific crane scheduling problem:

Layout: When setting up an end-of-aisle parts-to-picker system, the main decisions to be taken are the sizing of the ASRS (mainly regarding the length and height of the racking aisle, see also Roodbergen and Vis, 2009), the sizing of the bin buffer along the horseshoe conveyor, and the layout of the picking workstation. Regarding the latter, it has to be decided whether a single order bin at a time is assembled or the picker has access to multiple parallel orders. The latter is advantageous, if multiple orders demand the same SKUs, so that SKU bin deliveries can be saved (Füßler and Boysen, 2019). Existing decision support for evaluating a specific layout with regard to its prospective throughput is based on simulation (Manzini et al., 2006; Andriansyah et al., 2010, 2014), queuing models (Park et al., 1999; Claeys et al., 2016; Tappia et al., 2019) or analytical closed-form expressions (Bozer and White, 1990; Foley et al., 2004). We presuppose a given system layout consisting of a bin buffer with a given bin capacity and a picking workstation that only processes one order bin at a time (see Fig. 1).

Storage assignment: Ever since the seminal paper of Hausman et al. (1976), the main storage assignment policies within an ASRS (namely, dedicated storage, random storage, closest open location, full-turnover-based storage, and class-based storage, for details see Roodbergen and Vis, 2009) have received considerable attention. These policies decide where to store each SKU bin in the aisle of our ASRS. In-depth reviews on this topic are provided by Roodbergen and Vis (2009) as well as Gagliardi et al. (2012). In this paper, we assume a dedicated storage policy, where each SKU bin is retrieved from and stored back into a given shelf position in the racks.

Crane scheduling: Erecting an ASRS is costly; Roodbergen and Vis (2009), for instance, report that an aisle costs $634,000. Moreover, ASRS are based on plenty fixed hardware, so that once erected they are barely scalable. Therefore, the ASRS is often the bottleneck resource in a parts-to-picker system (see also Füßler and Boysen, 2019). Hence, efficiently executing the storage and retrieval requests of the crane is of utmost importance. In this context, our paper addresses an efficient scheduling of the crane in the ASRS when executing a given set of storage and retrieval requests. Important contributions providing exact solution procedures with polynomial runtime in this field stem from Lee and Schaefer (1997), who optimally pair storage and retrieval requests to dual commands with a front-end IO point, and Van den Berg and Gademann (1999), who determine dual commands for a fixed sequence of storage requests and IO points at both ends of the aisle. The latter result is extended by Gharehgozli et al. (2017) for problems without given sequence of storage requests. Further polynomial-time algorithms for special cases have been developed, e.g., by Dooly and Lee (2008) and Gharehgozli et al. (2021). For a tabular overview of related crane scheduling problems and their complexity status, we refer to Baardman et al. (2021, Table 1). An extensive review on these and other crane scheduling problems in ASRS is provided by Boysen and Stephan (2016). These papers, however, neglect the interdependence of crane scheduling with the picking subsystem. Storage and retrieval requests cannot arbitrarily be combined to dual commands, but a retrieval request delivered to the picking workstation becomes a storage request after completion and passing the bin buffer. This interdependence is considered by Bozer and White (1996), Mahajan et al. (1998), Baardman and Roodbergen (2016) and Baardman et al. (2021), who are our predecessors to address the end-of-aisle crane scheduling problem. Simple heuristics, such as nearest neighbor, are presented by Bozer and White (1996) and Mahajan et al. (1998). They, however, only address systems with a buffer capacity of two bins. Larger bin buffers of arbitrary given size are investigated by Baardman and Roodbergen (2016) and Baardman et al. (2021), who also extend the problem by considering waiting times. They show that the problem without waiting times is a special case of the multiple traveling salesman problem (mTSP) and propose an exact branch-and-cut approach as well as a simulated annealing heuristic tailored to the scheduling problem. With these procedures, they are able to solve instances with up to 40 requests and a bin buffer of 5. We build up on these predecessors, take up the problem setting without waiting times defined there, and contribute by settling the computational complexity, leveraging a reformulation as a VRP to use state-of-the-art solution techniques that allow us to solve instances with hundreds of requests within a few seconds or at most minutes, and deriving insights into the design of end-of-aisle systems with regard to the bin buffer.

Dwell point: Locating the dwell point where the crane waits for its next request only occurs during idle time. Important contributions to minimize the response time when relocating the crane toward a new request that just popped up are provided by Egbelu (1991), Egbelu and Wu (1993), Bozer and White (1984) and Van den Berg (2002). In our setting, the ASRS is the bottleneck resource of the parts-to-picker system, so that no idle time occurs. Instead, we assume a given set of SKU bins that already wait to be delivered toward the picking workstation and to be returned into the ASRS once the picks are completed.

Order processing: The order bins and their SKU demands to be satisfied in the end-of-aisle picking workstation heavily impact the sequence of SKU bins to be retrieved from the ASRS. If a current order demands pieces of multiple different SKUs, then the corresponding SKU bins should be delivered in direct succession in order to quickly completing this order. Intermixing this SKU delivery sequence with other (currently not demanded) SKUs, only produces additional crane effort without contributing to order completion. Furthermore, if a picking workstation has multiple storage slots for processing multiple order bins in parallel, then batching orders demanding the same SKUs can reduce the number of SKU bin deliveries. An optimization approach for this problem is provided by Füßler and Boysen (2019). Similar batch synchronization problems for other parts-to-picker system are provided by Füßler and Boysen (2017) and Boysen et al. (2017) for put-to-light and shelf-lifting mobile robots systems, respectively. The end-of-aisle crane scheduling problem only considers single-SKU orders where each pick list exclusively consists of a single order line. These orders build the vast majority in B2C online sales, where customer households tend to order few pieces per order (Boysen et al., 2019). Bansal et al. (2021), for instance, report on real-world order data from a large e-commerce retailer, where more than 95% of all orders follow this structure. Single-SKU orders allow that the SKU bins required by a given order set can be brought to the picking workstation in arbitrary order. Note that, since we follow the problem definition of our predecessors, this assumption is also contained in their research but not explicitly elaborated there.

3. Problem definition

This section defines the end-of-aisle crane scheduling problem, whose underlying setup is based on the following components. Note that, for convenience, the mathematical notation is summarized in Table 1.

Table 1
Notation for the end-of-aisle crane scheduling problem.

Sets	
S_0	Set of SKU bins in the buffer at the beginning of the planning run, $S_0 = \{s_1, \dots, s_b\}$
S	Set of SKU bins to be retrieved, $S = \{s_{b+1}, \dots, s_{b+n}\}$
Parameters	
b	Size (in number of SKU bins) of the buffer
n	Number of SKU bins to be retrieved
$d_{s,s'}$	Travel distance of the crane between the storage positions associated with SKU bins $s, s' \in S_0 \cup S$
Variables	
π	Sequence with $b+n$ elements, defining the order in which the SKU bins in $S_0 \cup S$ are delivered to the picking station; $\pi_j \in S_0 \cup S$ is the SKU bin at sequence position j

- *SKU supply subsystem:* We have a single crane that operates in a single aisle with racks for SKU bins to the left and right. The crane operates dual commands when retrieving homogeneous SKU bins all containing multiple pieces of a specific SKU from their dedicated storage positions and returning them back there after picking.
- *Picking subsystem:* We have a single end-of-aisle picking workstation, where a single human picker removes demanded pieces from SKU bins and puts them into subsequently active order bins. The workstation only processes single-SKU orders, so that SKU bins can be processed in any sequence.
- A *bin buffer* interconnects both previous subsystems. A horseshoe conveyor constitutes a first-in-first-out buffer with a fixed number of b slots that are subsequently passed. One of these slots is the storage position in the picking workstation, and when reaching the final slot the bin is the next to be returned into the ASRS by the crane.

Initially, we have a set $S_0 = \{s_1, \dots, s_b\}$ of b SKU bins that have already been retrieved from the ASRS and are left over from the preceding planning run in the bin buffer. Derived from the given set of picking orders, we also have an additional set $S = \{s_{b+1}, \dots, s_{b+n}\}$ of n SKU bins to be retrieved from the ASRS and delivered toward the picking workstation during the current planning run. The last b bins of the current planning run then, again, are the leftovers S_0 for the next planning run, and so on. The crane operates dual commands when delivering SKU bins, so that the retrieval of a SKU bin in the jth dual command is combined with the storage request of the SKU bin delivered during the $(j-b)$th dual command. Since we have a queue based first-in-first-out bin buffer, each newly arrived SKU bin becomes the bin to be returned back into the storage subsystem after successively moving through the b buffer positions.

In this setting, a solution is represented by a *SKU sequence* π that defines the sequence in which the $b+n$ SKU bins of $S_0 \cup S$ are delivered toward the picking workstation. Note that π_j defines the SKU bin delivered at sequence position j. The first b sequence positions are fixed with those given SKU bins already delivered during a previous planing run and to be returned into the ASRS. Furthermore, we have n free sequence positions, where a permutation of $S = \{s_{b+1}, \dots, s_{b+n}\}$ defines the delivery sequence of all demanded SKUs. Note that such a permutation ensures that each demanded SKU bin is delivered exactly once and that each sequence position receives exactly one SKU bin.

Given this decision context, we aim to minimize the empty movement of the crane when moving empty during a dual command between the storage of a bin returned into the ASRS and the retrieval of the next SKU bin summarized over all dual commands processed by the crane. The loaded movement, where the crane transports a SKU bin either as a storage or retrieval request, are inevitable once the sets of SKU bins S_0 and S are defined. Each of these bins has to be retrieved from its dedicated storage position and returned to this position exactly once, so that this part of the crane's workload is fixed and cannot be influenced by SKU sequence π. Thus, only the empty crane movement connecting the pairs of storage and retrieval request per dual command can be influenced. However, storage and retrieval requests cannot arbitrarily be paired. Due to the queue based first-in-first-out bin buffer, it is always the storage move of SKU bin π_j delivered within the $(j-b)$th dual command to be paired with SKU bin π_{j+b} retrieved during the jth dual command ($j = 1, \dots, n$). Thus, we seek a SKU sequence π, which minimizes

$$Z(\pi) = \sum_{j=1}^{n} d_{\pi_j, \pi_{j+b}}, \tag{1}$$

where $d_{s,s'}$ defines the travel distance of the crane between the storage positions dedicated to SKUs s and s'. We dub this problem EOA-CSP (end-of-aisle crane scheduling problem). Note that if potential savings of loaded crane travel were considered in our objective function, unattractive SKUs from the rearmost storage positions would be left in our bin buffer at the end of a SKU sequence. Their return, however, cannot be avoided but is only deferred to the next planning run where the leftover SKU bins constitute the initial bin buffer content S_0 of the next instance. Thus, over the long run, nothing can be saved, and we exclude potential savings of loaded crane travel in the objective function. Further note that crane travel in an ASRS typically follows a Chebyshev (maximum) metric, i.e., the distance between two points (x_0, y_0) and (x_1, y_1) in the plane is measured as $\max\{|x_0 - x_1|, |y_0 - y_1|\}$. For ASRS, this is often appropriate because crane and shuttle are propelled by separate engines and can move simultaneously (see Boysen and Stephan, 2016).

Example: Consider the example of Fig. 2, where set $S = \{A, E, G, K\}$ of SKU bins is to be retrieved from an ASRS that stores all SKUs A to L in dedicated storage positions. The single end-of-aisle picking workstation is fed by a bin buffer with $b = 3$ slots, which is initially filled with SKU bins $S_0 = \{C, D, F\}$ left over from the previous planning run. They have to be returned into the ASRS in the sequence $\langle C, F, D]$. We measure distances as the numbers of shelf spaces passed by the crane according to the Chebyshev metric. Solution 1 represents SKU sequence $\pi^1 = \langle C, F, D, A, K, G, E]$ and leads to an empty crane travel, indicated by the red arrows, of $Z(\pi^1) = 2 + 1 + 1 + 1 = 5$. Solution 2, instead, services SKU bins in sequence $\pi^2 = \langle C, F, D, K, G, A, E]$, which leads to an objective value of $Z(\pi^2) = 2 + 1 + 3 + 2 = 8$.

4. Analysis of computational complexity

In this section, we show that EOA-CSP constitutes a complex optimization problem and is strongly NP-hard. Note that Baardman and Roodbergen (2016) and Baardman et al. (2021) show that EOA-CSP can be reduced to a special case of the multiple traveling salesman problem (mTSP). This, however, is not sufficient to settle its complexity status, because it only demonstrates that the mTSP is at least as hard as the EOA-CSP (but not vice versa), which does not allow drawing conclusions as to the EOA-CSP's complexity.

Lemma 1. *EOA-CSP with buffer size $b = 1$ constitutes a special case of the path version of the traveling salesman problem (TSP).*

Proof. For the most restricted case with buffer size $b = 1$, EOA-CSP reduces to finding a minimum-distance schedule that starts with

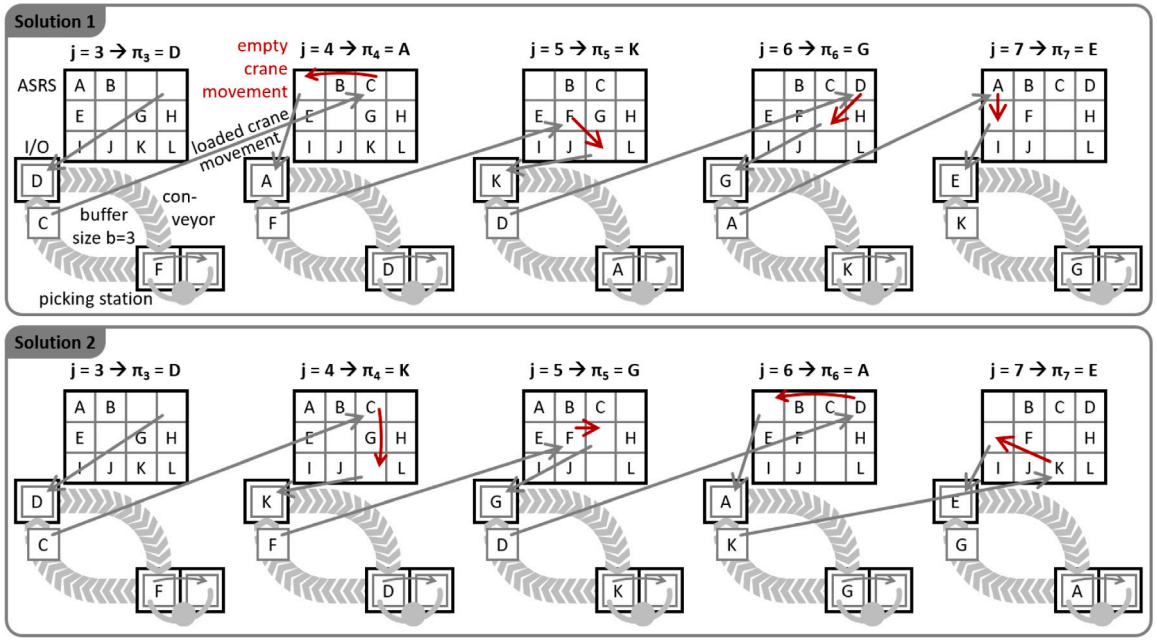

Fig. 2. Example for our end-of-aisle crane scheduling problem (EOA-CSP).

fixed SKU bin π_1 and processes all retrieval requests $j \in S_0 \cup S = \{s_1, s_2, \dots, s_{n+1}\}$ exactly once. Directly after the retrieval of SKU bin π_i $(i = 1, \dots, n)$, the crane proceeds with storing SKU π_i back into the ASRS again. This leads to an empty movement from the storage location of storage request π_i to the storage location of next retrieval π_{i+1} during a dual command. Optimizing the empty movement is then equivalent to solving the path version of a TSP with a given start location. Thereby, the travel distances $d_{s,s'}$ between the locations $s, s' \in \{1, 2, \dots, n+1\}$ to be visited follow the two-dimensional Chebyshev (maximum) metric. □

Without loss of generality, we can assume all storage locations to be visited having integer coordinates. EOA-CSP with buffer size $b = 1$ is, thus, very similar to the path version of the geometric TSP (G-TSP) with Manhattan distances.

G-TSP (path version) with Manhattan distances: Given a set P of points with integer coordinates in the plane and two specified points $s, t \in P$. Among all paths that start in s, visit every point in P exactly once, and end in t, find one that has the minimum total length according to Manhattan distances $d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$ for all $(x_1, y_1), (x_2, y_2) \in P$.

The only differences between G-TSP with Manhattan distances and EOA-CSP with $b = 1$ are the following:

- EOA-CSP employs the Chebyshev metric $d((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|; |y_1 - y_2|\}$ instead of the Manhattan metric, and
- EOA-CSP does not presuppose a predefined endpoint of the crane schedule.

By proving the following theorem, we show that the strong NP-hardness result for G-TSP (see Papadimitriou (1977)) carries over to EOA-CSP with $b = 1$.

Theorem 1. *EOA-CSP is strongly NP-hard, even if the buffer size is $b = 1$.*

Proof. Papadimitriou (1977) proves that the tour version as well as the path version (i.e., with a given start and a given end point) of G-TSP is strongly NP-hard when applying either discretized Euclidean distances $d((x_1, y_1), (x_2, y_2)) = \lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \rceil$ or Manhattan distances $d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$ for all $(x_1, y_1), (x_2, y_2) \in P$. The proof remains valid when applying Chebyshev distances $d((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2|; |y_1 - y_2|\}$ instead, since the Chebyshev and the Manhattan metric are equivalent under a 45°-rotation (see Garey and Johnson, 1979). Moreover, in the transformation scheme of Papadimitriou (1977) for the path version of G-TSP there is exactly one further node besides the given start point of the path that has only one node in its direct neighborhood. This further node is the only candidate node for being the end point of the path. Consequently, even allowing for an arbitrary endpoint of the path does not alter the complexity status of the path version of G-TSP. Thus, EOA-CSP is strongly NP-hard even if $b = 1$. □

Regarding the impact of the buffer size, Baardman et al. (2021) show that any instance of EOA-CSP with any arbitrary buffer size $b \geq 1$ can be transformed to an instance of the *multiple traveling salesmen problem with equal visits* (mTSPEV). While that alone does not settle the complexity status of EOA-CSP, Theorem 1 demonstrates NP-hardness of EOA-CSP with buffer size $b \geq 1$, as this is a generalization of the $b = 1$ case. Note that in edge cases, if buffers are very large, EOA-CSP becomes tractable. Specifically, if the buffer size equals the number of requests ($b = n$), EOA-CSP becomes polynomially solvable as a matching problem. We discuss the consequences of this further in our numerical study in Section 6.

5. EOA-CSP is a special case of the vehicle routing problem

Our complexity analysis shows that EOA-CSP constitutes a challenging optimization problem. In this section, however, we show that we do not have to start the development of tailor-made heuristics and exact algorithms like our predecessors in this field. Instead, building on the work of Baardman et al. (2021), we prove that EOA-CSP is actually equivalent to a special case of the well-know vehicle routing problem (VRP). This allows us to stand on the shoulder of giants and solve EOA-CSP instances with the large arsenal of VRP solution methods derived by previous research.

Lemma 2. *For each $k \in \{0, \dots, b-1\}$, the empty crane movement included in all dual commands c_j with $j \mod b = k$ jointly build a path.*

Proof. Due to the queue based first-in-first-out bin buffer of our part-to-picker system, it is always the storage move of SKU bin π_j

(i.e., delivered during dual command c_{j-b}) to be combined with the retrieval move of SKU bin π_{j+b} to dual command c_j (for all $j = 1, \dots, n$). Thus, after storing SKU bin π_j, the crane executes an empty crane move from the storage position of π_j to the storage position of SKU π_{j+b} and proceeds with the retrieval of π_{j+b}. After storing SKU bin π_{j+b} again (i.e., in dual command c_{j+b}), the crane executes an empty crane move from the storage position of π_{j+b} to the storage position of SKU π_{j+2b} and proceeds with the retrieval of π_{j+2b} (for all $j = 1, \dots, n-b$). The empty crane move included in c_{j+b} (for $j = 1, \dots, n-b$), thus, starts where the empty crane move included in c_j ends. This, however, allows for a merging of the empty crane movements contained in all dual commands c_j with $j \in \{1, \dots, n\}$ and $j \mod b = k$ for a $k \in \{0, \dots, b-1\}$ to one path. □

In the following, we show that EOA-CSP with buffer size b is a special case of a variant of the capacitated VRP. We dub this variant VRP*. Specifically, VRP* is defined as follows.

VRP*: Given are a set $C = \{1, 2, \dots, |C|\}$ of cities, a set $V = \{1, 2, \dots, |V|\}$ of vehicles, and distances $d(c, c') \in Z^+$ for each pair of cities $c, c' \in C$. Moreover, for each vehicle $v \in V$, there is given a capacity $w_v \in Z^+$ and an initial start location $s_v \in C$. The task consists in defining exactly one open route (i.e., without a final return to the start location) per vehicle in such a way that

- all cities $c \in C$ are contained in a route,
- each vehicle $v \in V$ starts its route at s_v,
- the number of cities visited by vehicle v is not larger than w_v for all $v \in V$, and
- the total length of all routes is minimum among all solutions meeting the previous constraints.

Note that the limited numbers of cities per route directly translate into capacity constraints of the vehicles.

Theorem 2. *EOA-CSP with buffer size b is a special case of VRP* with b vehicles moving according to the Chebyshev metric in the plane.*

Proof. According to Lemma 2, a solution to EOA-CSP with buffer size b decomposes into b paths of empty crane movements. $1+(n-1) \mod b$ of these paths consist of $\lceil \frac{n}{b} \rceil$ arcs and in the case of $n \mod b \neq 0$ there are $b-1-(n-1) \mod b$ paths consisting of $\lfloor \frac{n}{b} \rfloor$ arcs. $\pi_1, \pi_2, \dots, \pi_b$ are the starting points of these paths, whereas $\pi_1, \pi_2, \dots, \pi_{1+(n-1) \mod b}$ belong to paths consisting of $\lceil \frac{n}{b} \rceil$ arcs. Next, we describe how EOA-CSP translates into VRP*.

The set of storage positions $\{\pi_1, \pi_2, \dots, \pi_{n+b}\}$ and the b paths of EOA-CSP translate into the set of cities C and the routes for b vehicles $V = \{1, 2, \dots, b\}$ of VRP*, respectively. Thereby, $\pi_1, \pi_2, \dots, \pi_b$ of EOA-CSP serve as starting locations $s_1, s_2, \dots, s_b$ of VRP*, respectively. Finally, the maximum number of cities in routes $1, 2, \dots, 1 + (n-1) \mod b$ of EOA-CSP is $\lceil \frac{n}{b} \rceil + 1$ and the maximum number of cities in the remaining routes $V \setminus \{1, 2, \dots, 1+(n-1) \mod b\}$ is $\lfloor \frac{n}{b} \rfloor + 1$. Note that in a feasible solution to the resulting VRP* instance, all routes must have exactly these maximum lengths, which allows for retranslating the solution of the VRP* instance into the solution of the EOA-CSP instance. Otherwise, it would not be possible to visit all cities. □

Example: Consider set $S = \{A, B, E, G, J, K, L\}$ of SKU bins to be retrieved from an ASRS. Again, our picking workstation is fed by a bin buffer with $b = 3$ slots, which is initially filled with SKU bins $S_0 = \{C, D, F\}$ to be returned into the ASRS in sequence $\langle C, F, D]$ (see Fig. 3). The solution represents SKU sequence $\pi = \langle C, F, D, A, K, G, E, B, L, J]$ and results in $b = 3$ paths $D-G-L$, $C-A-E-J$ and $F-K-B$ of empty crane moves. The empty crane travel in these paths equals the tour length of three truck tours that amount to 2, 4 and 3, respectively. Thus, the objective value of both EOA-CSP and its corresponding VRP amounts to $Z(\pi) = 9$.

Table 2
Parameters for instance generation.

Description	Values
Gross slot size	1.2×1.5 m
Rack size	50×16 slots
Crane's vertical speed	24 meters per minute
Crane's horizontal speed	80 meters per minute
Number of open requests n	60, 120, 180, 240
Buffer size b	1, 2, 3, 5, 10, 20, 30, 60, 90, 120

6. Computational study

Before we elaborate on the results of our computational tests in Section 6.3, we report on some preliminary work. First, Section 6.1 defines our test instances, and then Section 6.2 reports on the application of a state-of-the-art VRP solver and the competitors of our benchmark test.

6.1. Test data

To generate instances, we follow the scheme used in Gharehgozli et al. (2017, 2021), among others. Since we want to show that our VRP transformation allows to solve much larger instances than previous research, we switch to this generation scheme instead of applying those of our predecessors. We assume a unit-load ASRS with a storage rack with 50 columns by 16 rows of storage slots for standard pallets. Slot sizes, crane speeds, and other instance parameters are in Table 2. Given this data, it is possible to calculate the Chebyshev travel time between any two arbitrary slots in the rack. Note that, like the cited papers, we disregard acceleration and deceleration times. Moreover, we do not consider service times for (un-)loading the pallets because these times can be assumed to be constant for a given request set. Also note that, internally, we round all time values to tenths of a second.

From these 50×16 storage locations, we randomly draw either 60, 120, 180, or 240 requests for set S and an additional $b \in \{1, 2, 3, 5, 10, 20, 30, 60, 90, 120\}$ storage requests that are initially in the buffer S_0. Thus, in an instance with, e.g., $n = 60$ request we always have to execute 60 dual commands, irrespective of the buffer size. Note that, for industry application, very large buffer sizes (e.g., $b = 120$) may be impractical. Apart from the small buffers, we nonetheless also test these large buffer values to gain insights into the algorithmic performance of our solution approach in edge cases and the effect of (large) buffers on system performance.

For each number of requests, we generate 20 random instances, yielding a total of $4 \cdot 20 = 80$ sets of n requests, each of which is paired with ten different buffer sizes b. The instances as well as detailed numerical results can be downloaded from https://doi.org/10.5281/zenodo.4629978.

6.2. Benchmark solution methods

We compare four solution methods to solve our EOA-CSP: first come, first served (*FCFS*), a matching-based heuristic based on Lee and Schaefer (1997) (*MBH*), the simulated annealing approach proposed by Baardman et al. (2021), and a vehicle routing heuristic to solve the EOA-CSP as *VRP**.

As a *VRP heuristic*, we use HGS-CVRP (Vidal, 2022), a state-of-the-art open source implementation of the hybrid genetic search originally described in Vidal et al. (2012). As the authors remark (Vidal, 2022, p. 9), "HGS-CVRP stands as the leading metaheuristic in terms of solution quality and convergence speed". HGS-CVRP produces an average gap of merely 0.11% over the best known solutions of benchmark library CVRPlib, which contains instances with up to 1.000 customers (Vidal, 2022). We use the code downloaded from https://github.com/vidalt/HGS-CVRP.

HGS-CVRP is geared toward solving the classic capacitated VRP (CVRP). We transform an instance of VRP* to a CVRP instance by

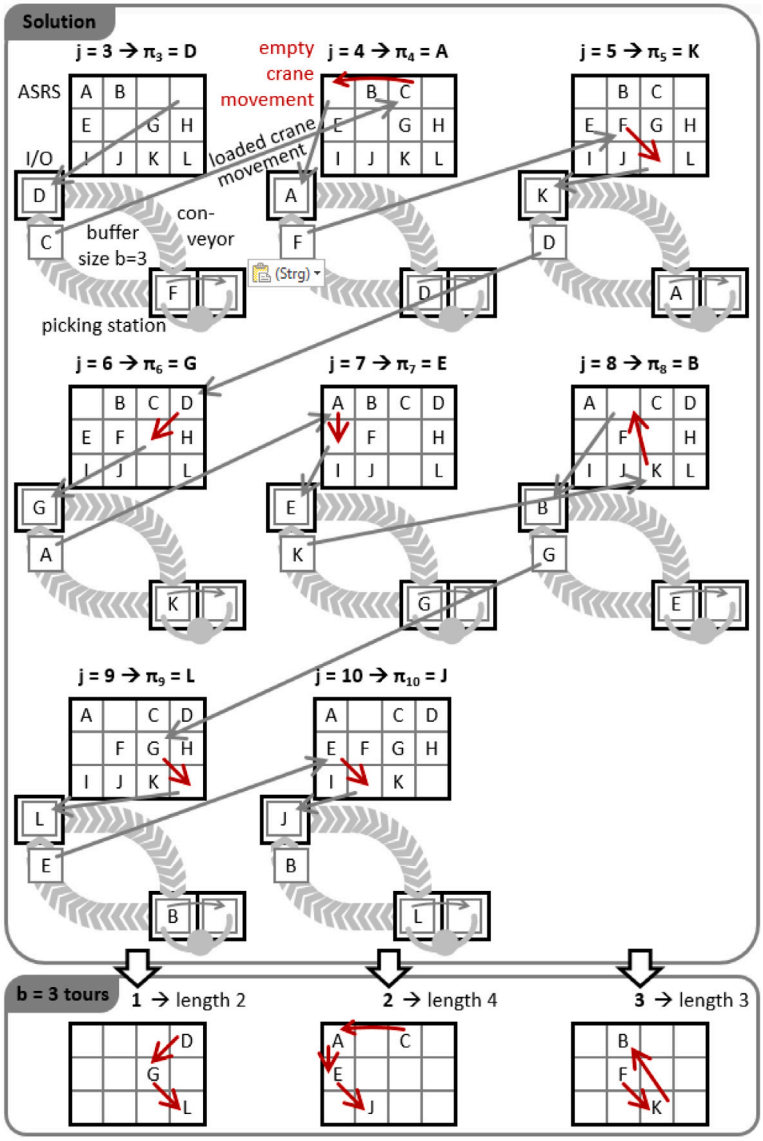

Fig. 3. Example for the equivalence of an EOA-CSP schedule and a capacitated VRP solution.

introducing a dummy depot v_0 to the set of vertices $V' = C \cup \{v_0\}$. Apart from the edges and weights already present in VRP*, we also add edges between v_0 and j, $\forall j \in S_0$, with weight $d(v_0, j) = d(j, v_0) = 0$. The distance $d(v_0, c)$, $\forall c \in C \setminus S_0$, is prohibitively large. Finally, the distance $d(c, v_0)$, $\forall c \in C$, is zero.

The above transformation creates a complete graph where all vehicles start from the same depot v_0. From there, they can only go to one of the vertices in S_0 (because the distances to any other vertices are prohibitively large), which stand for the requests that are initially in the buffer or, in VRP* terminology, they are the starting locations s_v. Subsequently, the vehicles can freely visit the other locations and, finally, return to the dummy depot without cost (because the distances from $c \in C$ to v_0 are zero).

In VRP*, the capacity demand at every location in C is 1. However, the CVRP presupposes homogeneous vehicles, while in VRP*, vehicles may have differing capacities. This can be adjusted by giving all vehicles the same capacity $\lceil \frac{n}{b} \rceil + 1$. Then, the first $1 + (n-1) \mod b$ vertices from ordered set S_0 have demand $1 + \lceil \frac{n}{b} \rceil - \lfloor \frac{n}{b} \rfloor$, effectively lowering the capacity of the vehicles passing through these vertices by 1 if n is not divisible by b. This ensures that all vehicles are loaded exactly to capacity, as in the original VRP*.

As a second benchmark algorithm to solve EOA-CSP, we apply the heuristic approach by Lee and Schaefer (1997), which is our first competitor from the existing literature. We refer to this procedure as the *matching-based heuristic* MBH, because it decomposes the overall problem setting into subsequent planning runs, where the current storage requests of the bin buffer are matched with the still remaining retrieval requests by solving an instance of the linear assignment problem. Specifically, MBH proceeds as follows:

1. Calculate the empty crane travel distance $\delta_{j,j'}$ for each request j in the buffer ($j \in S_0$) if it is followed by open request $j' \in S$. To get a square distance matrix, we add $|S| - |S_0|$ dummy requests that cause zero travel distance.
2. Solve a linear assignment problem to match the requests in the buffer with open requests such that the total empty travel time between the matched requests is minimal. Add the cost of the assignment to the objective value.
3. The matched open requests become the new buffered requests, replacing the requests in S_0. Let $S := S \setminus S_0$. If $|S| > 0$, go to step 4. Otherwise, end.

4. If $|S_0| > |S|$, i.e., there are fewer open requests left than there are requests in the buffer, consider only the first $|S|$ requests in S_0, i.e., $S_0 := \{s_1, \dots, s_{|S|}\}$. Go to step 1.

Note that the linear assignment problem can be solved in polynomial time; the classic improved Hungarian method, for instance, solves it in cubic time (Frank, 2005).

We also implement the simulated annealing approach (*SA*) originally proposed by Baardman et al. (2021). According to the computational study in that paper, SA is the most successful heuristic for EOA-CSP. We re-implement it as faithfully as possible. Note that while we do not use the exact same code as Baardman et al. (2021), given that, by design, SA runs through $O(n^5)$ iterations as well as the computational results published in the original article, the CPU times in our own numerical study are in line with expectations (see Section 6.3).

SA works on the sequence of requests π. Starting from a solution obtained via a nearest neighbor constructive heuristic, a neighborhood solution π' is reached by randomly swapping two requests in the incumbent sequence π, where the first b elements in the sequence are fixed because they are initially in the buffer. If the neighborhood solution is better than the incumbent, it is accepted. Otherwise, it is accepted with probability $e^{-(Z(\pi')-Z(\pi))/T(t)}$, where $T(t) = n^5/(1+e^{50 \cdot t/n^5-5})$ is the temperature in iteration $t = 1, \dots, n^5$. After a total of n^5 iterations, the best found solution is returned.

Finally, the *first come, first served (FCFS)* priority rule simply fills the first b positions in the sequence vector π with the given buffered requests S_0, and then adds the remaining requests in S sorted by their index (i.e., effectively randomly). FCFS emulates an approach where no optimization at all is applied to steer the order fulfillment process. Note that during their numerous site visits the authors met such un-optimized systems surprisingly often.

We implemented solution methods *FCFS*, *MBH*, and *SA* in C# 8.0. For VRP*, we used the publicly available C++ implementation of HGS-CVRP in its default settings (single-threaded execution, which stops after 20,000 iterations without improvement). All tests were run on an x64 PC with an AMD Ryzen 5 3600 CPU with a 3.6 GHz base clock and 32 GB of RAM.

6.3. Numerical results

This section elaborates the results of our computational study. Table 3 lists the overall results, averaged over the twenty instances per number of requests n. These results suggest the following findings:

- *Solution time:* The first thing to note is that the CPU times of both FCFS and MBH are negligible (at most a few milliseconds even for the largest instances) and hence not printed in the table. The SA from Baardman et al. (2021) takes about six minutes of CPU time per small instance ($n = 60$). On larger instances ($n \geq 120$), SA does not terminate within one hour; we therefore do not run these tests. This is in line with the findings of Baardman et al. (2021, p. 1167), who report that solving instances with $n = 120$ requests "takes on average 9218 s". HGS-CVRP working on VRP* can take a few minutes to solve the largest instances with $n = 240$, although the runtime does not exceed four minutes for any instance, which should still be acceptable for practical applications.
- *Solution quality:* We report the average total empty crane travel summarized over all dual commands per instance in minutes. First, we can conclude that not optimizing the dual commands and simply matching them in an FCFS manner seems not a good idea. Compared to our VRP*, FCFS produces between 4.5 times (i.e., for $n = 60$ requests and buffer size $b = 60$) and twelve times (i.e., for $n = 240$ and $b = 30$) more empty crane travel. The disadvantage of MBH compared to our VRP* crane scheduling is smaller yet still substantial (if $b < n$): it ranges between 23% (i.e., for $n = 120$ and $b = 1$) and 117% (i.e., for $n = 240$ and $b = 30$) additional empty crane travel. The SA from the literature exhibits an average gap of about 15% to MBH and about 55% to VRP* for $n = 60$. We do not test it on larger instances.

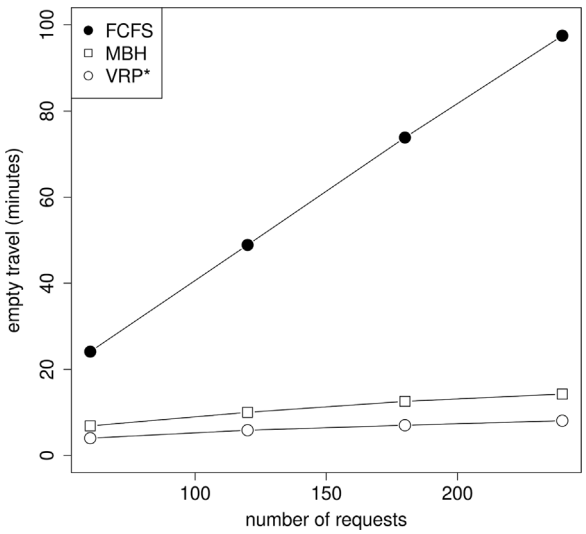

Fig. 4. Comparison of three approaches: first come, first served (FCFS), matching-based heuristic MBH, and our VRP* crane scheduling ($b = 10$).

We can conclude from theses tests that tailor-made heuristics of previous research cannot compete with the application of a state-of-the-art VRP solver. These results are confirmed by Fig. 4, which graphically compares FCFS, MBH, and VRP* for given buffer size $b = 10$. Clearly, simple FCFS scheduling is not appropriate at all for the EOA-CSP. Even the still relatively simple and fast MBH reduces empty crane travel by several orders of magnitude. Solving the VRP* reduces unproductive crane movement time even more, by about another 35% on average compared to the MBH solution.

Put another way, using a sophisticated optimization approach reduces unproductive crane travel by as much as about 1.5 h per 240 requests. As a comparison, note that the average loaded, productive travel time is about 1.2 min per request in our instances (including travel times for both retrieval and storage but disregarding service times). This implies that, using an optimized schedule, the crane could complete more than 310 requests in the same time in which a crane with a naive FCFS schedule could only complete 240 requests – a shortfall of 70 requests. The disadvantage of MBH is smaller, but our VRP* crane scheduling still allows up to eight additional requests.

An interesting thing to note in this context is that the objective values of the MBH and VRP* solutions match if $b = n$. This is because in these instances, every request in the buffer S_0 is matched with exactly one open request in S. A minimum cost matching (as per MBH) is optimal in this case. HGS-CVRP is capable of also finding this optimal solution, hence the objective values converge. Moreover, there seems to be a roughly linear relationship between the number of requests and the empty travel time. The reason is that more requests tend to add more travel time to the schedule simply because more locations have to be visited, although the increase is moderate for near-optimal schedules (VRP*).

Using VRP* schedules, Fig. 5 looks into the effect of the buffer size on empty crane travel. The graphs indicate that there is a sweet spot for the buffer size of about 10 to 20 bins, where the total empty crane travel is minimal. If the buffer is lower or greater, the crane utilization becomes worse. This can be explained by the fact that both very small and very large buffers limit the flexibility that the optimization approach has.

For instance, consider a problem with three requests $S = \{A, B, B'\}$, where the slots associated with requests A and B/B' are at opposite ends of the shelf, but requests B and B' are close to each other. Now assume the buffer size is $b = 1$ with $S_0 = \{A_0\}$ initially buffered. Assuming that request A_0 is close to request A (and hence far from

Table 3
Average total empty travel time of the crane in minutes.

# requests n	Buffer size b	FCFS obj.	MBH obj.	SA obj.	SA CPU s	VRP* obj.	VRP* CPU s
60	1	24.05	5.1	5.15	384.85	4.10	2.9
60	2	24.05	5.5	5.75	384.3	4.05	5.6
60	3	23.5	5.6	5.85	384.45	4.00	6.8
60	5	24.35	5.95	6.35	386.05	4.05	7.6
60	10	24.1	6.85	7.8	389.25	4.00	7.45
60	20	23.95	7.1	8.0	388.05	4.25	8.2
60	30	24.1	6.4	7.65	391.4	4.60	8.4
60	60	23.85	5.3	7.35	397.2	5.30	11.85
Avg.		23.99	5.98	6.74	388.19	4.29	7.35
120	1	48.35	7.4	–	–	6	6.45
120	2	49.1	7.8	–	–	6	16.75
120	3	49.4	8	–	–	6	22.4
120	5	48.75	8.95	–	–	5.9	22.25
120	10	48.9	10	–	–	5.85	19.85
120	20	49.1	11.95	–	–	6	19.4
120	30	48.95	11.65	–	–	6.3	21.5
120	60	48.4	9.95	–	–	6.6	21.6
120	90	49.2	10.1	–	–	6.8	28.3
120	120	48.4	8.25	–	–	8.25	34.25
Avg.		48.86	9.41	–	–	6.37	21.28
180	1	73.2	9.15	–	–	7.15	12.45
180	2	74.75	10.05	–	–	7.15	36.3
180	3	73.9	10.15	–	–	7.05	50.5
180	5	73.7	11.05	–	–	7.05	51.3
180	10	73.85	12.55	–	–	7	51.85
180	20	73.9	14.75	–	–	7.05	40.25
180	30	74.55	14.55	–	–	7.15	36.05
180	60	73.55	14.5	–	–	7.7	55.9
180	90	73.6	13.15	–	–	8.5	50.95
180	120	74.85	12.4	–	–	8.2	64.95
Avg.		73.99	12.23	–	–	7.40	45.05
240	1	98.3	10.45	–	–	8.25	21.95
240	2	98.15	10.9	–	–	8.15	76.55
240	3	97.3	11.5	–	–	8.1	84.9
240	5	98.25	12.45	–	–	8.15	114
240	10	97.5	14.25	–	–	8.05	90.2
240	20	97.35	17.35	–	–	8.15	76.25
240	30	99.2	17.7	–	–	8.15	64.5
240	60	97.8	17.85	–	–	8.7	76
240	90	98.25	15.95	–	–	8.75	83.5
240	120	98.85	14.55	–	–	9.4	91.25
Avg.		98.10	14.30	–	–	8.39	77.91

requests B and B'), the optimal processing sequence would be $\pi = \langle A_0, A, B, B' \rangle$ (or possibly $\pi = \langle A_0, A, B', B \rangle$), which causes a long empty crane move after storing bin A and before retrieving bin B (or B'). This cannot be avoided. If the buffer size is $b = 2$ with requests $S_0 = \{B_0, A_0\}$ initially buffered (where B_0 is close to B), it becomes possible to utilize the crane better: sequence $\pi = \langle B_0, A_0, B, A, B' \rangle$ avoids any long empty moves because the requests in both areas A and B/B' can be matched with suitable requests in the buffer. Finally, a buffer of size $b = 3$ with $S_0 = \{B_0, A_0, A_1\}$ (where A_0 and A_1 are close to A) again forces long empty crane moves because request A_1 cannot be matched with a nearby request.

Another way of looking at this is through the lens of the VRP*: Having a large buffer in EOA-CSP is tantamount to forcing the use of a large fleet of vehicles – possibly beyond what is optimal – in a routing problem. In the worst case, if the number of vehicles is equal to the number of customers (and every vehicle must be used, as is the case in the VRP*), each customer must be visited separately, which is almost always worse than combining at least a few nearby customers on a tour (note the classic savings algorithm, Clarke and Wright, 1964).

This illustrates why medium-size buffers tend to offer the most flexibility and best crane utilization. Note, however, that the choice of reasonable buffer sizes is quite wide. In all our instance groups, if the buffer size is somewhere between $b = 1$ and $b = 60$, the empty crane travel time fluctuates only by about a minute over the whole planning horizon. Given that there may be other issues that influence

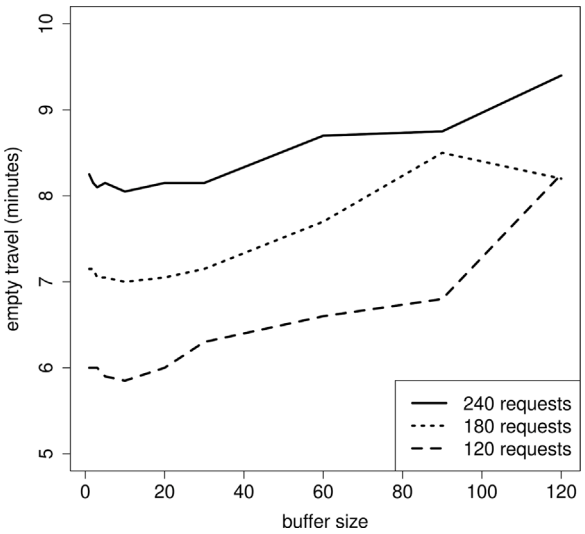

Fig. 5. Effect of buffer size on empty crane travel.

buffer choice, e.g., space constraints or the necessity to smooth over fluctuating pick times at the picking workstation, it may be helpful to know that a size of somewhere between one and sixty bins is likely to work well from the perspective of efficiently scheduling the crane.

7. Conclusions and future research

In this paper, we consider a crane scheduling problem that occurs in wide-spread parts-to-picker warehousing systems. Storage bins filled with SKUs have to be delivered from an ASRS to an end-of-aisle picking workstation and need to be returned back into the ASRS after picking. In this setting, the pairing of storage and retrieval requests to dual commands is interdependent, and we present an efficient solution method to solve this established crane scheduling problem. Specifically, we show that this problem, although it contains merely a single vehicle, namely, the crane, is equivalent to the capacitated VRP. This allows us to employ existing state-of-the-art algorithms of the rich arsenal of VRP solvers. Our computational study reveals the following take-home messages:

- After 30 years of research on end-of-aisle picking, the resulting crane scheduling problem is finally proven to be strongly NP-hard.
- Our transformation to the VRP allows to solve even large-sized instances with 240 retrieval requests and large buffers in a short time frame, substantially outperforming previous heuristics from the literature.
- Previous research offers only simple heuristics to solve the end-of-aisle crane scheduling problem. These tailor-made heuristics, however, cannot compete with the solution quality obtainable by a state-of-the-art VRP solver. This suggests that developing problem-specific solution methods of end-of-aisle picking problems may fundamentally not be worthwhile unless they offer a clear advantage over VRP tools.
- Not optimizing the processing of storage and retrieval requests, e.g., by processing them according to the FCFS principle, is costly. In the time a crane scheduled via FCFS operates 240 dual commands, a properly optimized system accomplishes 310.
- Mid-size buffers with a capacity for between ten and 20 bins perform best within our computational study when it comes to optimizing ASRS throughput, but deviating from the best buffer size is not overly costly.

Future research should investigate extended problem settings. For instance, cranes with multiple shuttles as well as picking workstations that process multi-item orders and/or multiple parallel order bins require an adaption of the crane scheduling. Although the applied heuristic VRP solver, HGS-CVRP, works quite well for EOA-CSP in our tests, there are other VRP solvers that may be even better suited to VRP*. In particular, Polten and Emde (2022) have shown that it can be worthwhile to adapt an exact solver like VRPSolver (Pessoa et al., 2020) to scheduling problems, which may also constitute a good starting point for EOA-CSP. In the case of EOA-CSP, additional symmetry breaking constraints may be constructed exploiting the special structure of VRP*, e.g., that every vehicle visits almost the same number of nodes. Furthermore, larger systems with many aisles and multiple workstations, all interconnected via conveyors, are even more involved. In these systems, it must be considered that bins currently processed at one workstation are not available at other workstations. A holistic scheduling approach for these systems remains another challenging task for future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Nils Boysen thanks the German Science Foundation (DFG) for their generous support of this work by the grant "Routing of human and robotic pickers in modern distribution centers" (BO 3148/14-1).

Appendix. Why Baardman et al. (2021) present no valid NP-hardness proof

In this appendix, we elaborate on the need for a proper NP-hardness proof for EOA-CSP, which is provided by the paper on hand. Baardman et al. (2021) state in their Theorem 1 that the problem is equivalent to the multiple traveling salesman problem, which is equivalent to the uncapacitated vehicle routing problem, with the additional restriction that all salesmen visit an equal number of cities (dubbed mTSPEV) and conclude strong NP-hardness of EOA-CSP. Their proof of equivalence, however, is incorrect. In particular, they only show that every instance of EOA-CSP can polynomially be transformed into an instance of mTSPEV in such a way that the optimal objective values of both instances are equal. Instead, for a proper NP-hardness proof one would have to show the opposite direction, i.e., that every instance of mTSPEV has a counterpart in EOA-CSP. Otherwise, EOA-CSP cannot be proven to be at least as hard as mTSPEV. Moreover, for a proper NP-hardness proof, it is crucial to explicitly account for the Chebyshev nature of the distances in EOA-CSP. Consequently, the mTSPEV with arbitrary distances is not the right problem for developing a reduction scheme.

Example: Imagine an instance of mTSPEV with only $m = 1$ salesman (i.e., an instance of the traditional TSP) where all distances are either 1 or 2. Obviously, if this instance includes at least ten cities it is not possible to reasonably translate the mTSPEV distances of either 1 or 2 into distances of EOA-CSP when it follows the infinity norm. Note that the TSP remains strongly NP-hard if all distances are either 1 or 2 (see Garey and Johnson, 1979). Thus, there are intractable instances of mTSPEV that have no counterpart in EOA-CSP.

It can be concluded that the paper of Baardman et al. (2021) lacks a valid complexity proof and only our paper finally settles the strongly NP-hard nature of EOA-CSP.

References

Allied Market Research, 2020. Automated storage and retrieval system market insights. URL https://www.alliedmarketresearch.com/automated-storage-and-retrieval-system-market-A06282. Last accessed June 30, 2021.

Andriansyah, R., Etman, L., Adan, I.J., Rooda, J.E., 2014. Design and analysis of an automated order-picking workstation. J. Simul. 8 (2), 151–163.

Andriansyah, R., Etman, L., Rooda, J., 2010. Flow time prediction for a single-server order picking workstation using aggregate process times. Int. J. Adv. Syst. Meas. 3 (1–2), 35–47.

Azadeh, K., De Koster, R., Roy, D., 2019. Robotized and automated warehouse systems: review and recent developments. Transp. Sci. 53 (4), 917–945.

Baardman, L., Roodbergen, K.J., 2016. Job sequencing in a miniload system. In: 14th IMHRC Proceedings (Karlsruhe, Germany – 2016).

Baardman, L., Roodbergen, K., J. Carlo, H., Schrotenboer, A., 2021. A special case of the multiple traveling salesmen problem in end-of-aisle picking systems. Transp. Sci. 55 (5), 1151–1169.

Bansal, V., Roy, D., Pazour, J.A., 2021. Performance analysis of batching decisions in waveless order release environments for E-commerce stock-to-picker order fulfillment. Int. Trans. Oper. Res. 28 (4), 1787–1820.

Boysen, N., Briskorn, D., Emde, S., 2017. Parts-to-picker based order processing in a rack-moving mobile robots environment. European J. Oper. Res. 262 (2), 550–562.

Boysen, N., De Koster, R., Weidinger, F., 2019. Warehousing in the E-commerce era: A survey. European J. Oper. Res. 277 (2), 396–411.

Boysen, N., Stephan, K., 2016. A survey on single crane scheduling in automated storage/retrieval systems. European J. Oper. Res. 254 (3), 691–704.

Bozer, Y.A., White, J.A., 1984. Travel-time models for automated storage/retrieval systems. IIE Trans. 16 (4), 329–338.

Bozer, Y.A., White, J.A., 1990. Design and performance models for end-of-aisle order picking systems. Manage. Sci. 36 (7), 852–866.

Bozer, Y.A., White, J.A., 1996. A generalized design and performance analysis model for end-of-aisle order-picking systems. IIE Trans. 28 (4), 271–280.

Claeys, D., Adan, I., Boxma, O., 2016. Stochastic bounds for order flow times in parts-to-picker warehouses with remotely located order-picking workstations. European J. Oper. Res. 254 (3), 895–906.
Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. 12 (4), 568–581.
Dallari, F., Marchet, G., Melacini, M., 2009. Design of order picking system. Int. J. Adv. Manuf. Technol. 42 (1–2), 1–12.
De Koster, R., 2007. Warehouse assessment in a single tour. In: Facility Logistics. Auerbach Publications, pp. 53–74.
De Koster, R., Le-Duc, T., Roodbergen, K.J., 2007. Design and control of warehouse order picking: A literature review. European J. Oper. Res. 182 (2), 481–501.
Dooly, D.R., Lee, H.F., 2008. A shift-based sequencing method for twin-shuttle automated storage and retrieval systems. IIE Trans. 40 (6), 586–594.
Egbelu, P.J., 1991. Framework for dynamic positioning of storage/retrieval machines in an automated storage/retrieval system. Int. J. Prod. Res. 29 (1), 17–37.
Egbelu, P., Wu, C., 1993. A comparison of dwell point rules in an automated storage/retrieval system. Int. J. Prod. Res. 31 (11), 2515–2530.
Foley, R.D., Hackman, S.T., Park, B.C., 2004. Back-of-the-envelope miniload throughput bounds and approximations. IIE Trans. 36 (3), 279–285.
Frank, A., 2005. On Kuhn's Hungarian method – a tribute from Hungary. Nav. Res. Logist. 52 (1), 2–5.
Füßler, D., Boysen, N., 2017. Efficient order processing in an inverse order picking system. Comput. Oper. Res. 88, 150–160.
Füßler, D., Boysen, N., 2019. High-performance order processing in picking workstations. EURO J. Transp. Logist. 8 (1), 65–90.
Gagliardi, J.-P., Renaud, J., Ruiz, A., 2012. Models for automated storage and retrieval systems: A literature review. Int. J. Prod. Res. 50 (24), 7110–7125.
Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company.
Gharehgozli, A., Xu, C., Zhang, W., 2021. High multiplicity asymmetric traveling salesman problem with feedback vertex set and its application to storage/retrieval system. European J. Oper. Res. 289 (2), 495–507.
Gharehgozli, A.H., Yu, Y., Zhang, X., De Koster, R., 2017. Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system. Transp. Sci. 51 (1), 19–33.
Hausman, W.H., Schwarz, L.B., Graves, S.C., 1976. Optimal storage assignment in automatic warehousing systems. Manage. Sci. 22 (6), 629–638.
Lee, H.F., Schaefer, S.K., 1997. Sequencing methods for automated storage and retrieval systems with dedicated storage. Comput. Ind. Eng. 32 (2), 351–362.
Litvak, N., Vlasiou, M., 2010. A survey on performance analysis of warehouse carousel systems. Stat. Neerl. 64 (4), 401–447.
Mahajan, S., Rao, B., Peters, B., 1998. A retrieval sequencing heuristic for miniload end-of-aisle automated storage/retrieval systems. Int. J. Prod. Res. 36 (6), 1715–1731.
Manzini, R., Gamberi, M., Regattieri, A., 2006. Design and control of an AS/RS. Int. J. Adv. Manuf. Technol. 28 (7–8), 766–774.
Papadimitriou, C.H., 1977. The Euclidean travelling salesman problem is NP-complete. Theoret. Comput. Sci. 4 (3), 237–244.
Park, B.C., Frazelle, E.H., White, J.A., 1999. Buffer sizing models for end-of-aisle order picking systems. IIE Trans. 31 (1), 31–38.
Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2020. A generic exact solver for vehicle routing and related problems. Math. Program. 183 (1), 483–523.
Polten, L., Emde, S., 2022. Multi-shuttle crane scheduling in automated storage and retrieval systems. European J. Oper. Res. 302 (3), 892–908.
Roodbergen, K.J., Vis, I.F., 2009. A survey of literature on automated storage and retrieval systems. European J. Oper. Res. 194 (2), 343–362.
Statista, 2019. eCOmmerce. URL https://www.statista.com/outlook/243/100/ecommerce/worldwide.
Tappia, E., Roy, D., Melacini, M., De Koster, R., 2019. Integrated storage-order picking systems: technology, performance models, and design insights. European J. Oper. Res. 274 (3), 947–965.
Van den Berg, J.P., 2002. Analytic expressions for the optimal dwell point in an automated storage/retrieval system. Int. J. Prod. Econ. 76 (1), 13–25.
Van den Berg, J.P., Gademann, A.J.R.M., 1999. Optimal routing in an automated storage/retrieval system with dedicated storage. IIE Trans. 31 (5), 407–415.
Vidal, T., 2022. Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood. Comput. Oper. Res. 140, 105643.
Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. 60 (3), 611–624.
Zou, B., De Koster, R., Xu, X., 2018. Operating policies in robotic compact storage and retrieval systems. Transp. Sci. 52 (4), 788–811.