A Special Case of the Multiple Traveling Salesmen Problem in End-of-Aisle Picking Systems

Baardman, Lennart; Roodbergen, Kees Jan; Carlo, Héctor J.; Schrotenboer, Albert H.

Published in:
Transportation Science

DOI:
10.1287/trsc.2021.1075

Publication date:
2021

Cite as:
Baardman, L., Roodbergen, K.J., Carlo, H.J., Schrotenboer, A.H., A Special Case of the Multiple Traveling Salesmen Problem in End-of-aisle Picking Systems, *Transportation Science*, to appear.

A Special Case of the Multiple Traveling Salesmen Problem in End-of-aisle Picking Systems

Lennart Baardman
Stephen M. Ross School of Business, University of Michigan, Ann Arbor, MI 48109, United States, baardman@umich.edu

Kees Jan Roodbergen
Faculty of Economics and Business, Department of Operations, University of Groningen, 9700 AV Groningen, the Netherlands, k.j.roodbergen@rug.nl

Héctor J. Carlo
Industrial Engineering Department, University of Puerto Rico – Mayagüez, Mayagüez, 00680, Puerto Rico, hector.carlo@upr.edu

Albert H. Schrotenboer
School of Engineering, Operations, Planning, Accounting, and Control Group, Eindhoven University of Technology, 5600 MB Eindhoven, the Netherlands, a.h.schrotenboer@tue.nl

This study focuses on the problem of sequencing requests for an end-of-aisle automated storage and retrieval system (AS/RS), where each retrieved load must be returned to its earlier storage location after a worker has picked some products from the load. At the picking station, a buffer is maintained to absorb any fluctuations in speed between the worker and the storage/retrieval machine. We show that, under conditions, the problem of optimally sequencing the requests in this system with a buffer size of m loads forms a special case of the multiple Traveling Salesmen Problem (mTSP) where each salesman visits the same number of cities. Several interesting structural properties for the problem are mathematically shown. In addition, a branch-and-cut method and heuristics are proposed. Experimental results show that the proposed simulated annealing-based heuristic performs well in all circumstances, and significantly outperforms benchmark heuristics. For instances with negligible picking times for the worker, we show that this heuristic provides solutions that are on average within 1.8% from the optimal value.

Key words: Automated Storage/Retrieval Systems; Multiple Traveling Salesmen Problem; Scheduling; Warehousing

1. Introduction

The percentage of products being sold in the business-to-consumer e-commerce market is ever-increasing, which in turn is impacting the requirements for the design and operation of the warehouses involved. Fast response times and minimal picking effort are fundamental for performing

customer-centric operations at high service levels against low costs (Xu, Allgor, and Graves 2009, Boysen, de Koster, and Weidinger 2019). Especially during peak hours, which result from cut-off times for same-day or next-day delivery, performance requirements are high. It is therefore essential to design warehouse systems with this peak performance in mind. As order picking is the most time-critical process in warehouse operations, automation of the order-picking process is often considered (Guardian 2013, Azadeh, De Koster, and Roy 2019). Such automation typically focuses on reducing or eliminating walking and driving time by workers, since this is known to be the most time-consuming part of manual order-picking systems (Tompkins et al. 2010). The actual handling of products remains a task for workers in most implementations due to the difficulties in automating the handling of products in varying shapes and sizes. Many automated systems exist, with a wide variety of conceptual designs. We study the end-of-aisle picking system, which is significant due to its long existence and wide adoption. The system is a serious contender for automation projects in the context of e-commerce, also when compared to newer conceptual designs such as the KIVA system (Bozer and Aldarondo 2018).

An *end-of-aisle picking system* consists of a number of aisles with storage racks along both sides of each aisle, where products are stored in loads. Here, *load* refers to any product carrier, which, depending on the exact system at hand, can be a plastic bin or a pallet. We consider an *aisle-captive system*, where each aisle is served by one automated storage/retrieval machine (S/R), which moves on rails through the aisle to store loads in and retrieve loads from the storage racks. The S/R machine, aisle, and racks together are generally referred to as an *Automated Storage and Retrieval System* (AS/RS) (Weidinger, Boysen, and Briskorn 2018, Azadeh, De Koster, and Roy 2019). When plastic bins are used for storing products, it is also referred to as miniload system, or miniload AS/RS system (Park et al. 2003, Park, Foley, and Frazelle 2006, Jaghbeer, Hanson, and Johansson 2020). At the end of each aisle, there is a picking station where a worker takes products from the retrieved load based on customer orders. After the worker has picked the products, the load with the remainder of the products is returned to its storage location. A schematic drawing of an end-of-aisle picking system (top view) can be found in Figure 1.

The sequence of retrieving loads from the AS/RS dictates the sequence in which the worker processes the loads, which in turn dictates the sequence in which loads are returned to the AS/RS. The throughput of the entire system is therefore directly dependent on the proper sequencing of requests. Notably, the S/R and the worker do not work on the same load at the same time. Therefore, differences are likely to occur between the worker's processing time and the S/R machine's cycle time, which may cause the S/R to wait for the worker or vice versa. The picking station is equipped with a buffer to reduce such waiting times and consequential adverse effects on throughput. This buffer allows the S/R (worker) to continue with the next load independently of the worker

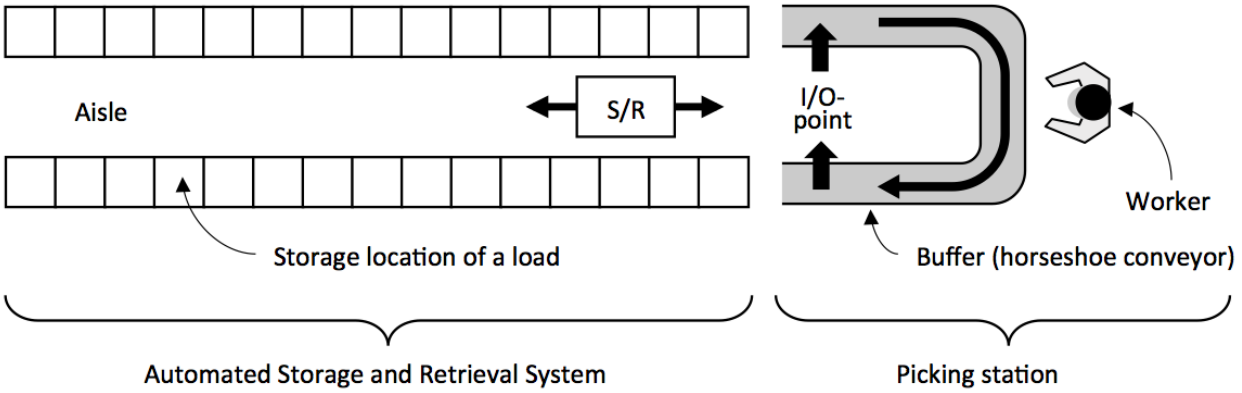

Figure 1 **Schematic drawing of an end-of-aisle picking system, consisting of a picking station and an Automated Storage and Retrieval System. Note that this figure gives a top view only, while the model also considers the second dimension (height) of the Automated Storage and Retrieval System.**

(S/R). Evidently, the limited capacity of the buffer sets bound to this, and waiting may still occur, albeit at a lower rate.

In this paper, we study the sequencing of requests in an end-of-aisle picking system. Specifically, we aim to maximize performance of this system during peak hours, by minimizing the makespan for handling all loads to be retrieved and stored. Therefore, we assume the warehouse management system regulates the overall process such that replenishments and prepositioning of loads are performed outside peak hours.

We study both the setting in which the S/R incurs no waiting time and the setting in which waiting times may occur. The setting without waiting times for the S/R may represent situations in which the buffer is sufficiently large or in which the worker's processing times are consistently lower than the cycle times of the S/R. For this setting, we prove that our end-of-aisle AS/RS request sequencing problem is mathematically equivalent to a special case of the multiple Traveling Salesmen Problem, mTSP (Bektas 2006). This equivalence is interesting for several reasons. First, it concerns a special case of the mTSP in which each salesman has to visit the same number of cities. This variant has not been studied before in literature, and we derive various structural properties for this problem. Second, we show that the number of salesmen in the mTSP formulation corresponds with the buffer size in the end-of-aisle picking system. The logic of this correspondence is best explained by defining and comparing the graphs of the two problems, which we describe in Section 4. A formal proof of the equivalence is provided in Appendix A.

The problem without waiting times is modeled as an integer program with a polynomial number of variables and constraints, which becomes computationally intractable for larger instances. Alternatively, we model it as an integer program with exponentially many constraints using a formulation inspired by the capacitated vehicle routing problem (Toth and Vigo 2002). The latter

approach combined with a branch-and-cut algorithm appears to be effective in solving instances to optimality within a three-hour time limit. Furthermore, we present several heuristics. Our simulated annealing heuristic provides near-optimal solutions in short computation times.

For the problem including waiting times, we define a recursive way to calculate the makespan for a given sequence of requests. We adapted all heuristics to cope with this nonlinear element, and we show that for varying pick times our heuristics outperform the heuristic that is based on the approach by Bozer and White (1996). Besides, we also test our heuristics' performance under a block-sequencing approach (Han et al. 1987), in which requests arrive dynamically. In this setting, our simulated annealing heuristic performs within 3% of the practically unattainable full information solution.

The remainder of this paper is structured as follows. First, a literature review is given in Section 2. Section 3 gives a description of the picking system studied. The equivalence of our end-of-aisle AS/RS request sequencing problem to the mTSPEV is shown in Section 4. Then, Section 5 presents the compact integer programming formulation for the mTSPEV, Section 6 presents several structural properties of the problem under consideration, and Section 7 describes a branch-and-cut method and four heuristics among which the simulated annealing heuristic. Lastly, Sections 8 and 9 present the experimental results and the conclusions, respectively.

2. Literature Review

The literature on AS/RS is extensive; Roodbergen and Vis (2009), Gagliardi, Renaud, and Ruiz (2012), and Boysen and Stephan (2016) give comprehensive literature reviews on the topic. Research focuses on the design and optimization of various aspects of warehouse systems including: (1) rack and aisle layout (Cardona and Gue 2020), (2) estimating performance measures by means of closed-form expressions or otherwise (Bozer and White 1996), (3) developing and evaluating storage assignment rules (Weidinger, Boysen, and Briskorn 2018, Weidinger and Boysen 2018), (4) finding a suitable dwell-point for the S/R (Van den Berg 2002), and (5) optimizing request sequencing (Gharehgozli et al. 2017).

In this paper, we specifically look at the problem of optimizing request sequences. This problem has also been studied in various other contexts. For example, Vis and Roodbergen (2009) consider the problem of scheduling container storages and retrievals in a container terminal. This problem is a special case of the directed Rural Postman Problem, and can be reformulated as an asymmetric Steiner Traveling Salesman Problem. Even though the general Steiner Traveling Salesman Problem is NP-hard, the special case considered in their paper can be solved in polynomial time. More recently, Yuan and Tang (2017) and Galle, Barnhart, and Jaillet (2018) also incorporate the aspect of relocation, next to the retrieval and storage, of containers or loads.

In the context of warehousing systems, Lee and Schaefer (1997) consider an AS/RS in a single aisle with one depot, where the lists of retrievals and storages are independent, and the order of the storages is pre-determined. Due to the problem's structural properties, both optimal and heuristic policies are developed and shown to reduce the travel time of the S/R significantly. In Gharehgozli et al. (2017), an AS/RS with two depots is considered. The authors establish properties of the problem that allow them to develop a polynomial-time algorithm. Dooly and Lee (2008), Popović, Vidović, and Bjelić (2014), Wauters et al. (2016), and Yang et al. (2017) consider the case where the S/R can carry multiple products. In Dooly and Lee (2008), an efficient polynomial-time algorithm can be developed due to the use of shift-sequencing, while NP-hardness forces Popović, Vidović, and Bjelić (2014) to resort to a nearest neighborhood heuristic and genetic algorithm. Wauters et al. (2016) develop an effective branch-and-bound procedure and propose a fast metaheuristic for larger instances. Due to NP-hardness, Yang et al. (2017) develop heuristics to simultaneously decide on the location of a load and the sequencing of orders. In another setting, Yu and De Koster (2012) study a multi-deep AS/RS, in which the S/R picks from a rack where loads are stored multiple layers deep. As the problem is NP-hard, they develop new heuristics to achieve near-optimal order sequencing. Recently, several papers propose heuristics for order sequencing in puzzle-based storage systems (Gue et al. 2014, Mirzaei, De Koster, and Zaerpour 2017, Yalcin, Koberstein, and Schocke 2019), which are compact systems in which each load can be retrieved independently without the use of an aisle. The AS/RS request sequencing problem that we study is equivalent to a special type of multiple Traveling Salesman Problem (mTSP), which is also NP-hard (see Section 4). We characterize several properties that make our problem distinctive from the standard mTSP (see Section 6). In line with literature for related problems, which are NP-hard like our problem, we use heuristics, metaheuristics, and branch-and-cut to solve our problem.

The type of end-of-aisle picking system that we study has been considered by several papers before. For example, Bozer and White (1990) and Foley, Hackman, and Park (2004) developed performance estimates for similar systems. Bozer and White (1996) give methods to determine the required number of pickers, whereas Park, Frazelle, and White (1999) develop methods to determine an appropriate buffer size. However, to the best of our knowledge, only two papers studied the sequencing of requests in our setting. In Mahajan, Rao, and Peters (1998) a nearest neighbor heuristic is presented for sequencing requests in an end-of-aisle picking system where loads must be returned to their original location. Bozer and White (1996) also present heuristics for this problem. Both articles, however, only study systems with a buffer size of 2. Modern systems provide more flexibility for handling loads at the picking station, facilitating the accommodation of more loads in the buffer. Therefore, we study systems where the buffer can have any size. For

Table 1 Overview of related work on AS/RS request sequencing.

	system characteristics				Solution methods			
	system type	waiting times	buffer size	re-stock	NP-hard	greedy heuristic	meta-heuristic	exact method
Tanaka and Araki (2009)	$[\mathrm{E}\|\mathrm{IO}^2, \mathrm{open}\|C_{\max}]$	-	-	-	-	-	-	-
Yu and De Koster (2012)	$[\mathrm{F}\|\mathrm{IO}^2, \mathrm{open}, \mathrm{FCFS}\|C_{\max}\|]$	-	-	-	✓	✓	-	-
Popović, Vidović, and Bjelić (2014)	$[F, 3\|\mathrm{IO}^2, \mathrm{class}\|C_{\max}]$	-	-	-	-	✓	✓	-
Yang et al. (2015)	$[F, k\|\mathrm{IO}^2, \mathrm{open}\|C_{\max}]$	-	-	-	✓	✓	✓	-
Wauters et al. (2016)	$[B, 2\|\mathrm{M}^2, \mathrm{zone}, \mathrm{prec}\|\sum w_i(T_i)]$	-	-	-	-	✓	✓	-
Nia, Haleh, and Saghaei (2017)	$[\mathrm{F}\|\mathrm{IO}^2, \mathrm{dy}, \mathrm{FCFS}\|*]$	-	-	-	✓	-	✓	-
Man et al. (2021)	$[E^{\mathrm{free}}\|\mathrm{IO}^2\|C_{\max}, \sum w_i(T_i)]$	-	-	-	-	-	✓	-
Lee and Schaefer (1997)	$[\mathrm{F}\|\mathrm{IO}^2\|C_{\max}]$	-	-	-	p	✓	-	✓
Van Den Berg and Gademann (1999)	$[\mathrm{E}\|\mathrm{IO}^2, \mathrm{FCFS}\|C_{\max}]$	-	-	-	p	✓	-	✓
Dooly and Lee (2008)	$[F, 2\|\mathrm{IO}^2\|C_{\max}]$	-	-	-	p	✓	-	✓
Gharehgozli et al. (2017)	$[E^{\mathrm{free}}\|\mathrm{IO}^2\|C_{\max}]$	-	-	-	p	✓	-	✓
Bozer and White (1996)	$[\mathrm{F}\|\mathrm{IO}^2\|C_{\max}]$	✓	2	✓	-	✓	-	-
Mahajan, Rao, and Peters (1998)	$[\mathrm{F}\|\mathrm{IO}^2\|C_{\max}]$	-	2	✓	✓	✓	-	-
This Paper	$[\mathbf{F}\|\mathbf{IO^2}\|\mathbf{C}_{\max}]$	✓	**m**	✓	✓	✓	✓	✓

benchmarking purposes, we compare our solution methods to generalized versions of heuristics presented by Bozer and White (1996) and Mahajan, Rao, and Peters (1998).

In Table 1, we present an overview of AS/RS request sequencing literature. Here we use the classification of Boysen and Stephan (2016) to indicate the system configuration in the column "system type". Note that since Boysen and Stephan (2016, Section 5) consider our setting an "unexplored case", their classification scheme does not capture all relevant aspects for the system we study. Hence we add three columns with system characteristics to this table to capture properties of models that also encompass the picking activity. The system characteristic "waiting times" indicates whether waiting times for the crane and the picker are taken into account in the scheduling algorithm, "buffer size" indicates the number of retrieved loads that can reside outside the aisle of the AS/RS to wait for the worker or to undergo picking, "restock" indicates that the storage location of a load is dedicated and equals the location from which the load was retrieved in an earlier cycle. In terms of solution approaches, "NP-hard" indicates whether the problem studied is NP-hard ('p' a polynomial-time algorithm is provided, '✓' NP-hardness is asserted in the paper, '-' paper does not specify computational complexity), "greedy heuristic" indicates whether constructive heuristic(s) without backtracking have been presented, "metaheuristic" indicates whether heuristic constructs have been used that allow for reconsidering earlier decisions, and "exact method" indicates whether an algorithm is presented (beyond off-the-shelf/commercial solvers) for solving the problem to optimality.

In Section 4, we show that the end-of-aisle AS/RS request sequencing problem is a special case of the multiple Traveling Salesmen Problem (mTSP) where each salesman visits the same number of

cities. The standard Traveling Salesman Problem (TSP), as described in Laporte (1992), concerns the sequencing of visits to a set of locations (or cities) such that the total travel distance incurred in visiting all locations exactly once and returning to the departing location is minimized. The TSP can be generalized into the multiple Traveling Salesmen Problem (mTSP), as described in Bektas (2006), by sharing the work between m salesmen instead of a single salesman. Bellmore and Hong (1974) describe how the standard mTSP can be mathematically transformed into the standard TSP. It should also be noted that the standard mTSP can be considered as a relaxation of the Vehicle Routing Problem with the capacity restrictions removed (Bektas 2006).

The literature includes several optimal algorithms for the mTSP (Gavish and Srikanth 1986, Gromicho, Paixão, and Branco 1992, Laporte and Norbert 1980, Svestka and Huckfeldt 1973, Uit het Broek et al. 2020) and multiple mathematical formulations (see Sarin et al. (2014) for a comparison of 32 formulations.) There are some heuristic solutions for the mTSP, mainly based on partitioning a TSP tour into m subtours (Russell 1977, Potvin, Lapalme, and Rousseau 1989, Frederickson, Hecht, and Kim 1978). However, the majority of the mTSP literature uses metaheuristics to solve the problem. Carter and Ragsdale (2006), Yuan et al. (2013), Singh and Baghel (2009), Sedighpour and Yousefikhoshbakht (2012), Tang et al. (2000), and Yu et al. (1997) use Genetic Algorithm-based heuristics. Larki and Yousefikhoshbakht (2014) propose an Evolutionary Algorithm-based heuristic. Other heuristics used for the mTSP include Artificial Bee Colony (Venkatesh and Singh 2015); Ant Colony (Liu et al. 2009, Yousefikhoshbakht, Didehvar, and Rahmati 2013, Ghafurian and Javadian 2011), General Variable Neighborhood Search (Soylu 2015), Neural Networks (Hsu, Tsai, and Chen 1991, Modares, Somhom, and Enkawa 1999, Somhom, Modares, and Enkawa 1999, Wacholder, Han, and Mann 1989), Simulated Annealing (Song, Lee, and Lee 2003), Tabu Search (Golden, Laporte, and Taillard 1997), and Gravitational Emulation (Rostami et al. 2015). Since the problem discussed in this paper is a special case of the mTSP (where each salesman visiting the same number of cities) we expect the existing mTSP heuristics to be ineffective for our problem as they would generate mostly infeasible solutions. In Section 7 we describe the method of Soylu (2015) for mTSP, which we adapted to ensure feasible solutions. We compare our proposed simulated annealing-based heuristic with results from the heuristic of Soylu (2015) to demonstrate our method's effectiveness

3. System Description

As introduced above, we are studying an Automated Storage and Retrieval System (AS/RS) with a picking station at the end of the aisle. A schematic drawing of the system, including the main terminology of system components, is given in Figure 1. Orders of customers arrive during the day. Each order may generate one or more requests for loads to be retrieved from the system. A *retrieval*

request results in a specific load being moved by the storage/retrieval machine (S/R) from its rack position to the picking station. At the picking station, a worker takes the required products from the load. After picking from the load, a *storage request* is issued for the load, where the S/R picks up the load from the picking station and puts it back in the rack.

AS/RSs are said to operate either in single-command or dual-command mode. In single-command mode, the S/R performs only a storage request or a retrieval request at each time. In dual-command mode, the S/R starts at the I/O-point, picks up a storage request, puts the load at its designated storage location, moves to the load that must be retrieved, and brings it to the I/O-point. The I/O-point of the AS/RS is where the S/R picks up and drops off the loads, i.e., it is located at the picking station. Typically, the single-command mode is used when there are either only storage requests or only retrieval requests available. Notice that when operating in single-command mode, the total completion time to handle all requests is the same regardless of the sequence in which the loads are handled. However, if operating in dual-command mode, the total completion time to handle all requests depends on the storage and retrieval sequence. For the dual-command mode, in the base case when storage and retrieval requests are known and independent, the problem of sequencing all requests can be written as a linear assignment problem, which is solvable in polynomial time (Lee and Schaefer 1997). In addition, Ascheuer, Grötschel, and Abdel-Hamid (1999) considers online sequencing of independent storage and retrieval requests.

A buffer is placed at the picking station to accommodate differences in speed between the S/R and the worker. Several configurations for the buffer exist, including the ones studied in Bozer and White (1990) and Park, Frazelle, and White (1999). A typical configuration is the horse-shoe configuration, which has a small conveyor system at the picking station in the shape of a horse shoe. Loads are dropped off by the S/R at one end of the conveyor and picked up from the other end. In-between, the worker retrieves products from the retrieved loads. From a modeling point of view, the essence is that these systems have a buffer at the picking station, and that loads go through the buffer in a strict First-In-First-Out manner. We assume the number of loads in the buffer to be constant. This is true by default when performing dual-command cycles since a new retrieval request accompanies every storage request. Consequently, for a buffer size of m loads, any retrieved load will appear as a storage request exactly m cycles after it was retrieved. This is consistent with the assumptions in Bozer and White (1996) and Mahajan, Rao, and Peters (1998), who studied a setting with a fixed buffer size of 2.

We are interested in the performance of the system during peak hours. In systems with a fairly constant utilization, storage location assignments of loads can be improved on the fly by returning retrieved loads to another location than the one it was originally retrieved from (Han et al. 1987), which may be beneficial in some situations (Carlo and Giraldo 2012). In contrast, we assume that

the warehouse management system has made the best possible effort to prepare the system for peak hours during the off-peak period. This implies that during peak hours, there is no need to reposition loads to better locations. All loads are assumed to be already stored at their respective best possible locations, based on information about the upcoming retrieval requests or a forecast thereof, see Chen, Langevin, and Riopel (2011) or Gu, Goetschalckx, and McGinnis (2007). Therefore, retrieved loads are best returned to their original storage location from which they were earlier retrieved. This assumption is consistent with Bozer and White (1996) and Mahajan, Rao, and Peters (1998). We further work from the base assumption that the buffer starts and ends empty. This assumption is realistic in practice for environments where completion times of sets of requests are formed such that they coincide with break times. The assumption of empty buffers is also helpful in the mathematical exposition. However, it is not essential to our solution approach. In Section 5 we indicate how the model can be easily adapted to include non-empty starting or ending conditions. In Section 8.4, we perform a block sequencing experiment that incorporates these conditions explicitly.

For any scheduling method, we need a set of requests that can be scheduled. However, requests do not arrive in sets but rather arrive dynamically one-by-one during the day. The general method for addressing this discrepancy is the *block sequencing* approach (Han et al. 1987). In block sequencing, all incoming requests are initially queued in the computer system until a condition is met, for example, based on a time stamp, a specified maximum number of unscheduled requests, or (approaching) completion of the current set of requests. Then, a set of requests is released from the queue for scheduling. Based on this, except for the block sequencing experiment in Section 7.5, we assume that we have a known set of requests that need to be executed.

Our paper's main objective is to minimize the makespan for handling all requested loads during peak hours. Indirectly, this also translates into a new system's purchase price because the required maximum achievable throughput of a system is an essential factor in the system's price. Essentially, this objective is the same as maximizing worker utilization as used in Bozer and White (1996) since worker utilization can be determined as total picking time divided by makespan. Firstly, makespan is influenced by total travel time or distance (Roodbergen and Vis 2009). S/R machines can simultaneously travel horizontally and vertically as they use two independent motors. Therefore, S/R travel is best described as Chebyshev travel (or infinity norm), where travel distance (or time) is determined as the maximum of the horizontal and the vertical distance (or time). Secondly, makespan is influenced by waiting times. These waiting times occur in situations where the buffer is full and the worker has not finished picking from any of the loads in the buffer. Since the S/R cannot complete its assigned work until the worker has retrieved products from all loads, minimizing the S/R's makespan also minimizes the worker's makespan.

Summarizing, we study the problem of minimizing makespan by sequencing requests for an S/R in an end-of-aisle picking system with a picking station buffer of a given size, where each retrieved load is to be put back at the same location from which it was retrieved. In the next section, we first show the relation of our AS/RS request sequencing problem to the mTSP.

4. A Special Case of the mTSP

This section shows that the end-of-aisle AS/RS request sequencing problem is equivalent to a special case of the multiple Traveling Salesmen Problem (mTSP). The mTSP extends the well-known Traveling Salesman Problem by considering multiple salesmen that each have a lower and upper bound on the number of visits in the tour. The special case of the mTSP that we consider in this paper is obtained by adding the additional constraint that each salesman must visit the same number of cities as the other salesmen. The intuition why the end-of-aisle AS/RS request sequencing problem is equivalent to this mTSP variant comes from the fact that -in a sense- each position in the buffer can be considered to be replenished by another salesman in an alternating fashion. So, for example, in a system with a buffer size of 3, 'salesman 1' delivers the first, fourth, seventh, ... load, while 'salesman 2' delivers the second, fifth, eighth, ... load, and 'salesman 3' delivers the third, sixth, ninth, ... load. When 'salesman 1' delivers the fourth load, this load replaces the buffer's first load. This first load is then returned to storage by 'salesman 1', who also originally had retrieved this load. Thus, each 'salesman' continuously updates one position of the buffer. In the actual system, there is only one S/R that transports all loads. However, the resulting travel path of this S/R can be mathematically decomposed into these separate paths of multiple salesmen, as we demonstrate in the remainder of this section.

It must be noted that the equivalence only holds if the S/R does not incur any waiting times. Otherwise, the special case of the mTSP provides a lower bound to the end-of-aisle AS/RS request sequencing problem. The probability of incurring waiting times depends on the interplay between several factors, such as the speed of the S/R, the worker's speed, the set of requests, the request scheduling, and the size of the buffer. Mathematically, zero waiting times for the S/R are obtained only with absolute certainty if picking times are set to zero. However, it is possible to mitigate S/R waiting times by increasing the buffer size. In the next section, we introduce a general formulation for the makespan that includes waiting times, and both formulations with and without waiting times are considered in the results section.

For the remainder of this section, we assume zero waiting time of the S/R so that the makespan of the S/R equals its total travel time. We first formalize descriptions of both problems and then show the correspondence between the two systems. We use the following notation for the end-of-aisle AS/RS request sequencing problem:

m: Size of the buffer at the picking station,

n: Number of loads to retrieve (and store),

v_i^r: Location in the rack for the retrieval request of load $i = 1, ..., n$,

v_i^s: Location in the rack for the storage request of load $i = 1, ..., n$,

v_0: Location of the I/O-point.

Notice that each load i will be handled twice; once to retrieve the load and once to store the load after picking has occurred at the picking station. These two events occur at different times but correspond to the same load and the same physical storage location in the system. Notice that a storage request of load i means that load i needs to be put back at the location from where it was retrieved earlier as retrieval request i. Nevertheless, when we construct the network, v_i^r and v_i^s will have different arcs incident to them, so we need to separate notation.

We can see our problem as an arc routing problem on a digraph $G = (V, A)$. The vertex set is given by $V = \{v_0, v_1^r, ..., v_n^r, v_1^s, ..., v_n^s\}$. The arc set A consists of *required arcs* for retrievals, (v_i^r, v_0), $i = 1, ..., n$, required arcs for storages, (v_0, v_i^s), $i = 1, ..., n$, and all remaining arcs that make a complete graph serve as *non-required arcs*. A route will make numerous visits to the I/O-point to pick up or drop off loads; it is required to visit the I/O-point before every storage to pick up the load that needs to be stored, and to visit the I/O-point after every retrieval to drop off the load that was retrieved.

We now first define the special case of the mTSP, before showing the correspondence between the two problems. We define the *multiple Traveling Salesmen Problem with Equal Visits (mTSPEV)* as the problem of finding a solution of minimum summed travel distance (or time) for routes of m salesmen, who all start and end at the depot, such that each city is visited exactly once, and such that each salesman visits exactly the same number of cities. We use the following notation for the mTSPEV:

m: Number of salesmen,

n: Number of cities to visit,

v_i: Vertex for city $i = 1, ..., n$,

v_0: The depot.

It is assumed that the number of cities is a multiple of the number of salesmen, i.e., $n = bm$, for some $b \in \mathbb{N}^+$. If violated, we can add $\lceil n/m \rceil m - n$ dummy cities (located at the depot) to satisfy this assumption without loss of generality. We use parameter m in the request sequencing problem for the size of the buffer, and we use the same m in the mTSPEV formulation for the number of salesmen. This is intentional, since these will be shown to correspond.

We note that the mTSPEV is an $\mathcal{NP}$-hard problem, as it is a generalization of the TSP, because any TSP can be written as it by simply taking $m = 1$. Additionally, the mTSPEV is equivalent to

the capacitated vehicle routing problem (CVRP) with the demand in each of the n cities equal to 1, and the capacity of each of the m vehicles equal to n/m.

THEOREM 1. *The end-of-aisle AS/RS request sequencing problem without waiting times is equivalent to the mTSPEV.*

Proof. See Appendix A.

In Section 7 of Bozer and White (1996) an explanation can be found that suggests the validity of the theorem for the case with $m = 2$. A formal proof for the general case is given in Appendix A. Here we show the validity of the theorem on an intuitive level and provide an example as illustration. Consider any given retrieval sequence for the S/R. For our example we use the following sequence of retrieving loads: 1-5-2-6-3-7-4-8. For the picker to receive loads in that order the S/R machine must perform a series of storage and retrieval requests. We denote the movements of the S/R machine by a set of tuples, with each tuple corresponding to a cycle of the S/R. We define a *cycle* as the combined movements of the S/R from leaving the I/O-point until the S/R's first following return to the I/O-point. Adjacent occurrences of v_0 are technically superfluous, however, they will ease the explanation. For our example, the tuple notation that describes the S/R movements for a system with a buffer capacity of 2 is as follows:

$$(v_0, v_1^r, v_0)(v_0, v_5^r, v_0)(v_0, v_1^s, v_2^r, v_0)(v_0, v_5^s, v_6^r, v_0)(v_0, v_2^s, v_3^r, v_0)...$$
$$...(v_0, v_6^s, v_7^r, v_0)(v_0, v_3^s, v_4^r, v_0)(v_0, v_7^s, v_8^r, v_0)(v_0, v_4^s, v_0)(v_0, v_8^s, v_0).$$

Figure 2(a) gives a graphical representation of the actual moves of the S/R machine for the example. The S/R starts at the I/O-point, moves to the storage location of load 1, picks up the load and brings it to the I/O-point, where the load enters the buffer. Next, the worker can take products from the load, after which the load is ready to leave the buffer and needs to await pickup by the S/R. While the worker is picking products from load 1, the S/R travels to retrieve load 5 from its location. After dropping off load 5 at the I/O-point, load 5 enters the buffer. Next, the S/R takes load 1 from the buffer and puts it back in its original storage location in the rack, the S/R moves empty to the location of load 2, and brings load 2 to the I/O-point. The retrievals of loads 1 and 5 are single-command cycles; the storage of load 1 together with the retrieval of load 2 forms a dual-command cycle. We assume the system to start and end with an empty buffer, however, this assumption is not restrictive as is shown in Section 5. Now, the following two observations illustrate the equivalence with the mTSPEV:

1. All required arcs are present in all feasible solutions, which implies that their combined length merely adds a constant to the objective value. Therefore we can remove the required arcs without affecting optimality. Figure 2(b) shows the resulting graph for the example situation. All arcs to or

from the I/O-point are required arcs, except for the first m arcs from the I/O-point to retrievals, and the last m arcs from storages to the I/O-point. In the example these are (v_0, v_1^r), (v_0, v_5^r), (v_4^s, v_0), and (v_8^s, v_0).

2. We can now merge each pair of vertices v_i^r, v_i^s into a single vertex v_i. This is permissible as they are physically at the same location at a distance of zero. This gives us m routes. Figure 2(c) depicts the result for our example.

Clearly, from Figure 2 we can see that an mTSPEV solution was obtained from an end-of-aisle AS/RS solution by applying the two steps above, neither of which alters optimality. This outline suggests, that there is a one-to-one correspondence between the two problems. The formal proof is given in Appendix A. That is, assuming no waiting times for the S/R, any solution to the end-of-aisle AS/RS request sequencing problem has an equivalent solution in the mTSPEV problem (and vice versa), and the objective values between the two solutions differ only by a constant. A notable consequence of this equivalence is that the request sequencing problem can be formulated without the use of precedence constraints. In its base description, the request sequencing problem requires each location v_i to be revisited exactly m moves after the first visit to that location. However, due to the reformulation to the mTSPEV, we can now see that the two visits to the same location are merged into a single node, with the intermediate visits contained in the routes of the other salesmen. Since the two visits are merged into a single node, it thus follows that it is not necessary to model the precedence relations explicitly.

5. Formulation

Given the results from the previous section, one can mathematically formulate the end-of-aisle AS/RS request sequencing problem without waiting times for the S/R as an integer linear program for a standard mTSP with an additional constraint for the number of visits. Similarly, one may adapt the mTSP formulation in Kara and Bektas (2006) by equaling the minimum and maximum number of cities to be visited by each salesman. Instead, we opt to present a mathematical formulation based on the operation of the end-of-aisle AS/RS request sequencing problem and then show how it may be interpreted for the mTSPEV. We first provide the integer linear programming model for the situation without waiting times. After that, we introduce formulations for waiting times of the S/R and the worker, as well as provide a formulation for the makespan.

The indices, parameters, and decision variables used in the model are defined as follows:

m: Size of the buffer at the picking station,

n: Number of loads to retrieve (and store),

i, j: Indices for the requested loads, $i, j = 1, \ldots, n+m$,

k: Index for cycles, $k = 1, \ldots, n+m$,

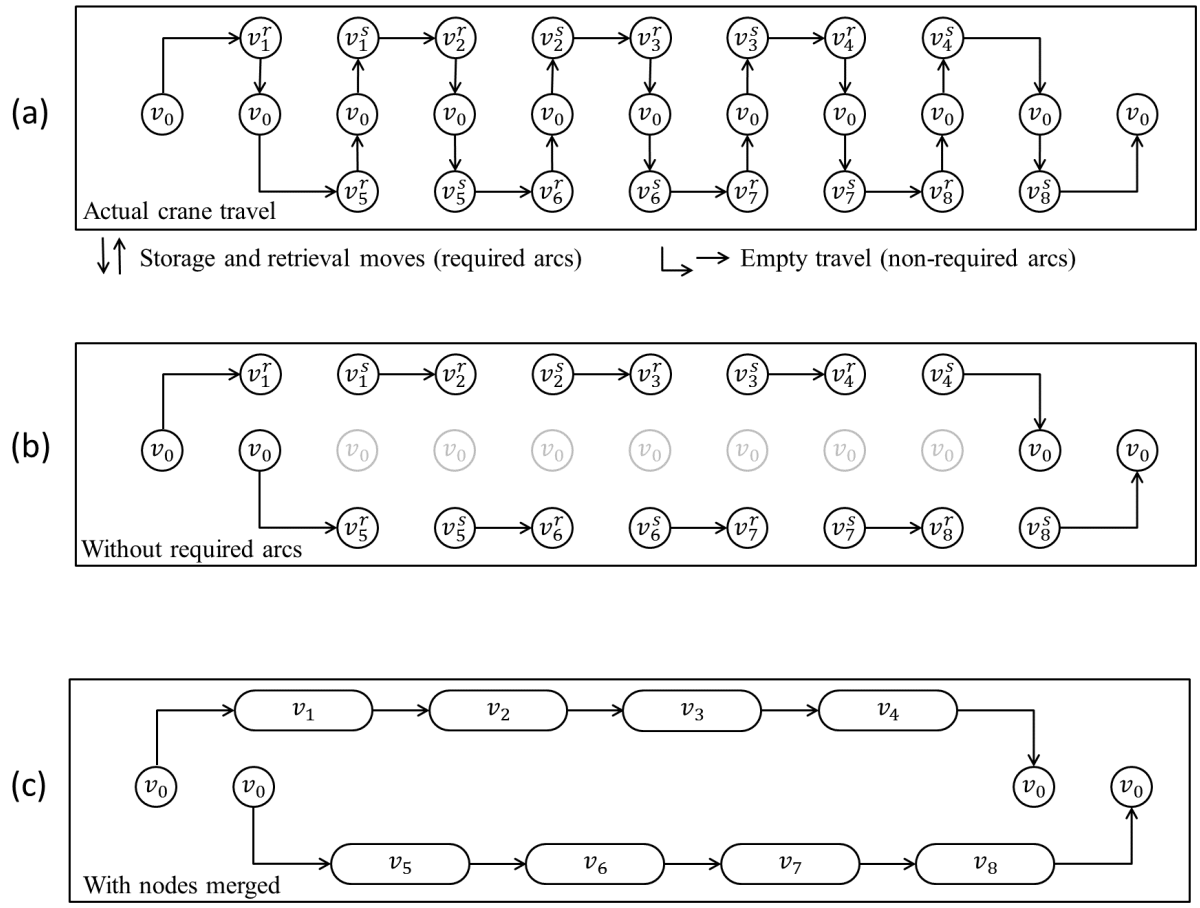

Figure 2 **Routing of the end-of-aisle AS/RS request sequencing problem, (a) all arcs traveled by the S/R, (b) excluding those arcs that the S/R is always required to travel, (c) merging the nodes that equate to retrieval and storage of the same load.**

d_{ij}: Distance (or time) traveled between the storage locations of loads i and j,

x_{ij}^k: $\begin{cases} 1 & \text{if load } i \text{ is stored and load } j \text{ is retrieved in cycle } k \\ 0 & \text{otherwise} \end{cases}$.

To explain the variable x_{ij}^k in our earlier graph notation, we can see $x_{ij}^k = 1$ to be equivalent to performing the tuple (v_0, v_i, v_j, v_0) in cycle k. Now, the model for the end-of-aisle AS/RS request sequencing problem without waiting times is given by:

$$\min \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} \sum_{k=1}^{n+m} d_{ij} x_{ij}^k \tag{1}$$

$$\text{such that } \sum_{j=1}^{n+m} \sum_{k=1}^{n+m} x_{ij}^k = 1 \qquad i = 1, \ldots, n+m \tag{2}$$

$$\sum_{i=1}^{n+m}\sum_{k=1}^{n+m} x_{ij}^{k} = 1 \qquad j = 1,\ldots,n+m \tag{3}$$

$$\sum_{i=n+1}^{n+m}\sum_{j=1}^{n} x_{ij}^{k} = 1 \qquad k = 1,\ldots,m \tag{4}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^{k} = 1 \qquad k = m+1,\ldots,n \tag{5}$$

$$\sum_{i=1}^{n}\sum_{j=n+1}^{n+m} x_{ij}^{k} = 1 \qquad k = n+1,\ldots,n+m \tag{6}$$

$$\sum_{i=1}^{n+m} x_{i\ell}^{k} = \sum_{j=1}^{n+m} x_{\ell j}^{k+m} \qquad k,\ell = 1,\ldots,n \tag{7}$$

$$x_{ij}^{k} \in \{0,1\} \qquad i,j,k = 1,...,n+m \tag{8}$$

The Objective Function (1) is to minimize the makespan, i.e., the total travel distance (or time) incurred to retrieve and store the requests. Equations (2) and (3) guarantee that each load is stored and retrieved exactly once, respectively. Equation (4) ensures that the first m retrievals are performed under single-command during the first m cycles (i.e., until the buffer is full). Note that Equation (4) is written in the same format as the "normal" dual-command cycles in Equation (5). However, in Equation (4), the storage requests are replaced by dummy jobs $n+1,...,n+m$ due to which essentially only single-command retrievals remain. Equation (5) ensures that the $n-m$ retrievals and storages under dual command are performed during the subsequent $n-m$ cycles. Equation (6) ensures that the last m storages are performed under single-command. Equation (7) guarantees that the load retrieved at cycle k should be stored at cycle $k+m$. Equation (8) defines the decision variables to be binary. Note that there are m dummy requests that also have to be "retrieved" and "stored", which act as I/O points. These dummy request are labeled with indices $i,j = n+1,...,n+m$, and are assumed to be located at the I/O-point.

Besides, consider the mTSPEV by having salesmen move in turns. During the first m turns, each salesman moves from the depot to its first destination. Then, during the next m turns, each salesman moves from its first destination to its second destination, and so on. Given this strategy for modeling the mTSPEV, the formulation of the end-of-aisle AS/RS request sequencing problem may be reinterpreted as follows. The objective function of the model (Equation 1) is to minimize the total travel distance (or time) incurred to travel to all cities. Equations (2) and (3) guarantee that each city is departed and visited exactly once, respectively. Equation (4) ensures that during the first m turns, each salesman moves from the depot towards its first destination (i.e., until all salesmen are at their first destination). Equation (5) ensures that during the subsequent $n-m$ turns each of the $n-m$ moves between cities are performed. Equation (6) ensures that during

the last m turns, each salesman moves from its last destination towards the depot (i.e., until all salesmen are at the depot). Equation (7) guarantees that the city visited by a salesman in turn k is left by the same salesman in turn $k+m$. Equation (8) defines the decision variables as binary.

In this model description, we assume the buffer to be empty at the start and at the end of a request sequence. As previously discussed, this may not be appropriate under a block sequencing approach. Nevertheless, this assumption may be easily relaxed. The loads that already reside in the buffer beforehand or the loads that remain in the buffer afterwards, simply replace the dummy jobs. Distances are adjusted as follows. Define d_{ij} for $i = n+1, \ldots, n+m$, $j = 1, \ldots, n$ as the distance from the storage location of load i that currently resides in the buffer to storage location j. And define $d_{ij} = 0$ for $i = 1, \ldots, n$, $j = n+1, \ldots, n+m$ since the last m retrievals will stay in the buffer.

5.1. Makespan with waiting times

The Objective Function (1), which is based on the mTSP, minimizes the makespan based on travel distances but does not consider S/R waiting (i.e., idle) times. S/R waiting times occur when the S/R becomes idle after delivering a load, but the picker has not finished picking the S/R's next load. In this subsection, we extend the Objective Function (1) to consider waiting times.

For ease of notation, we start by defining a permutation of the retrieval requests. The first retrieval of S/R is at the first position of the permutation, and the second retrieval is on the second position, and so on. When we use this permutation instead of the original indices, brackets are put around the appropriate indices. Based on this permutation, we define the following new variables and parameters. First, $W^c_{(k)}$ represents the waiting time of the S/R before it can perform the k'th cycle in the request sequence. Second, $W^p_{(k)}$ denotes the waiting time of the worker before being able to start picking from the load retrieved in the k'th cycle. The parameter $p_{(k)}$ represents the picking time associated with the load retrieved in the k'th cycle. Finally, we define travel times $d_{(k)}$ as the travel time involved in performing the k'th cycle. Now, for a given retrieval sequence, the S/R's waiting time before being able to start request cycle $k = 1, \ldots, n+m$ is given by:

$$W^c_{(k)} = \max\left\{ \sum_{t=1}^{k-m} (W^p_{(t)} + p_{(t)}) - \sum_{t=1}^{k-1} (d_{(t)} + W^c_{(t)}), 0 \right\}.$$

The first summation gives the total time spent by the worker waiting for the S/R and picking from the first $k-m$ loads (if $k-m \leq 0$, the summation is assumed to return a value of zero). The second term gives the total time spent by the S/R to perform cycles $1, ..., k-1$. Note that the S/R has to wait only if the S/R was faster at performing cycles $1, ..., k-1$ than the worker was at picking from loads $1, ..., k-m$.

Similarly, for a given retrieval sequence, the worker's waiting time for retrieval $k = 1, \ldots, n$ is given by:

$$W^p_{(k)} = \max\left\{\sum_{t=1}^{k}(d_{(t)} + W^c_{(t)}) - \sum_{t=1}^{k-1}(W^p_{(t)} + p_{(t)}), 0\right\}.$$

The makespan with inclusion of waiting times can now be written as:

$$\sum_{k=1}^{n+m}\left(d_{(k)} + W^c_{(k)}\right), \tag{9}$$

where the first term indicates the travel times of the S/R and the second term the cumulative waiting times of the S/R. The original objective function (Equation 1) without waiting times is straightforwardly obtained from Equation (9) by setting $W^c_{(k)} = 0$ for all k. Due to the fact that Equation (9) is a non-linear function, it cannot serve as an objective in a Mixed Integer Linear Programming formulation. However, it can be used in heuristics (see Section 7).

In order to illustrate these formulas, we revisit the example from Section 4. The request sequence of the example started with the cycles $(v_0, v^r_1, v_0)(v_0, v^r_5, v_0)(v_0, v^s_1, v^r_2, v_0)(v_0, v^s_5, v^r_6, v_0)$. For the first cycle (v_0, v^r_1, v_0) the S/R clearly does not need to wait before starting, so $W^c_{(1)} = 0$. The worker will have to wait for the S/R to retrieve the first load, hence $W^p_{(1)} = d_{(1)}$. After dropping off the first load at the I/O-point, the S/R can continue instantly with the second cycle, (v_0, v^r_5, v_0), since the buffer capacity is 2. So, there is no waiting time, $W^c_{(2)} = \max\{-d_{(1)}, 0\} = 0$. The worker only has to wait for the second load if it takes longer for the S/R to retrieve the second load than it takes for the worker to pick from the first load, $W^p_{(2)} = \max\{d_{(2)} - p_{(1)}, 0\}$. Next, the S/R should start on the third cycle, (v_0, v^s_1, v^r_2, v_0). This cycle can only start if the worker has finished picking from load 1. Hence, the S/R has to wait if the finish time of the worker (consisting of the worker's waiting time to start on load 1 plus the picking time for load 1) is later than the finish time of the S/R itself (consisting of the cycle times for loads 1 and 2), so $W^c_{(3)} = \max\{W^p_{(1)} + p_{(1)} - d_{(1)} - d_{(2)}, 0\}$. In the end, this process of interchangeably calculating the waiting times for the S/R and the worker continues for all cycles in the request sequence.

6. Structural Properties

This section elaborates on several interesting structural properties of the mTSPEV and its link with the mTSP. These properties emphasize the importance of theory development for the mTSPEV, since we demonstrate that solutions of the mTSPEV exhibit significantly different behavior from the solutions of the general mTSP. Before progressing, we introduce the following notation: let $L^{TSP}_m(I(V))$ and $L^{TSPEV}_m(I(V))$ denote the length of an optimal mTSP and mTSPEV route, respectively, for instance I with vertex set V and m salesmen. Furthermore, we assume that the distances (or times) between locations satisfy the triangle inequality.

First, we present a property that shows an important difference between the mTSP and mTSPEV. Whereas in the mTSP the length of an optimal route increases when the number of cities (n) increases, this is not necessarily true for the mTSPEV. The first property shows when the length of an optimal route of the mTSPEV can be decreased by adding a set of cities.

PROPERTY 1.

$$L_m^{TSP}(I(V)) < L_m^{TSPEV}(I(V)) \Leftrightarrow \exists U \text{ with } L_m^{TSPEV}(I(V \cup U)) < L_m^{TSPEV}(I(V)).$$

Proof. See Appendix B.

Thus, assuming that the triangle inequality holds and that the optimal mTSP route is shorter than the mTSPEV route through the set V, there exists a set of vertices U such that the length of the optimal mTSPEV route through the union of sets V and U is strictly shorter than through the set V alone.

The following corollary presents a case in which one can be assured that the optimal route length of an mTSPEV route reduces by increasing the number of vertices with a certain amount. This is the case when the optimal mTSP and mTSPEV route are not the same for a certain instance I with a vertex set V.

COROLLARY 1. *Assume $L_m^{TSP}(I(V)) < L_m^{TSPEV}(I(V))$. For $k = 1, \ldots, m$, let S_k be the set of vertices visited by salesman k in an optimal mTSP solution for instance $I(V)$. Define $s(k) = \max_i |S_i| - |S_k|$. Then, the vertex set $U = \{v_1^1, \ldots, v_{s(1)}^1\} \cup \{v_1^2, \ldots, v_{s(2)}^2\} \cup \cdots \cup \{v_1^m, \ldots, v_{s(m)}^m\}$ fulfills the requirements of Property 1, if there exist vertices $x, y \in S_k$ for which $d(x, y) = d(x, v_i^k) + d(v_i^k, y)$, for all $k = 1, \ldots, m$ and $i = 1, \ldots, s(k)$.*

Similarly, there is a difference in the behavior the mTSP and mTSPEV show in response to a change in the number of salesmen (m). This difference is formalized in the following property.

PROPERTY 2.

$$L_m^{TSP}(I(V)) \leq L_{m+1}^{TSP}(I(V)) \text{ for all } V \text{ and } m \in \mathbb{N}^+.$$
$$L_m^{TSPEV}(I(V)) > L_{m+1}^{TSPEV}(I(V)) \text{ for some } V \text{ and } m \in \mathbb{N}^+.$$

Proof. See Appendix C.

Hence, assuming the triangle inequality holds, the length of an optimal mTSP route always becomes longer or remains equal if the number of salesmen is increased. On the other hand, the length of an optimal mTSPEV route could reduce when the number of salesmen is increased.

The following property shows in which situations one can be assured that the optimal route length of an mTSPEV route strictly decreases for a decrease in the number of salesmen.

PROPERTY 3. *Assume the strict triangle inequality holds, then*

$$L_a^{TSPEV}(I(V)) < L_{ba}^{TSPEV}(I(V)) \text{ for all } V \text{ and } a, b \in \mathbb{N}^+, b \geq 2.$$

Proof. See Appendix D.

For the first problem, the number of salesmen equals a, while for the second problem they are a multiple of a. Intuitively, a move between two vertices should be shorter than moving from the depot to both vertices and thus when merging several salesmen's routes, the result should be shorter. This leads to the following corollary comparing the TSP and mTSPEV.

COROLLARY 2. *Assume the strict triangle inequality holds, then*

$$L_1^{TSP}(I(V)) = L_1^{TSPEV}(I(V)) < L_m^{TSPEV}(I(V)) \text{ for all } V \text{ and } m \in \mathbb{N}^+, m \geq 2.$$

Combining these results, we observe that the mTSPEV behaves substantially different from the mTSP or TSP. In fact, these properties illustrate that it is difficult to find structural patterns that could be exploited to develop a heuristic for the problem.

Finally, we deal with theoretical lower bounds of the mTSPEV. The formulation of the mTSPEV in Section 4 shows that it is a constrained Linear Assignment Problem (LAP). Therefore, the LAP is a lower bound to the mTSPEV, which can easily be solved in polynomial time. Another lower bound to the mTSPEV is the relaxed version of the mTSPEV formulation, where the values of the decision variables x_{ij}^k are not constrained to be binary, but to lie anywhere between 0 and 1. In Section 8 we will elaborate further on the latter lower bound.

7. Solution Methods

The model for the mTSPEV as presented in Section 5 can be solved via (commercial) MIP solvers such as Gurobi or CPLEX. However, our initial experiments demonstrated that a large percentage of instances with $n = 40$ could not be solved to optimality within 7.5 hours this way. To overcome this, we first discuss in this section an effective branch-and-cut algorithm to solve the MTSPEV to optimality.

Furthermore, since the mTSPEV has been shown to be $\mathcal{NP}$-hard in Section 4, we present four heuristic approaches to generate solutions for our end-of-aisle AS/RS request sequencing problem, both with and without waiting times. First, in Section 7.2, we present a nearest neighbor heuristic, which is a rule-of-thumb heuristic originally developed for the TSP and mTSP (Johnson and McGeoch 1997), and adapted for end-of-aisle AS/RS request sequencing by Mahajan, Rao, and Peters (1998). In Section 7.3, we present the second heuristic coming from Bozer and White (1996)

which is similar in nature to the first, but specifically accounts for waiting times. Both heuristics, from Mahajan, Rao, and Peters (1998) and Bozer and White (1996), are designed for sequencing requests in an end-of-aisle system with a buffer size of 2. Thus, we made minor adaptations to the heuristics so that they can handle any buffer size. The third approach, presented in Section 7.4, is a general variable neighborhood search heuristic based on the method of Soylu (2015) for solving the mTSP. This heuristic represents a state-of-the-art heuristic for solving the standard mTSP, which meant that some adaptations had to be made to the heuristic so that it would guarantee to provide feasible solutions for the mTSPEV. Finally, in Section 7.5, we propose a simulated annealing heuristic. As is demonstrated in Section 8, our simulated annealing heuristic provides high-quality solutions.

7.1. Branch-and-cut Algorithm

As is readily shown, the mTSPEV can be formulated as an mTSP with m routes with an equal number of visits. Suppose sufficient dummy nodes are included to ensure that n is an integer multiple of m. In that case, we can solve this problem as a capacitated vehicle routing problem where each city i has weight 1, the vehicle has capacity n/m, and where we set the number of outgoing arcs from the depot $\{n+1\}$ equal to m. That is,

$$\min \quad \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} d_{ij}y_{ij} \tag{10}$$

$$\text{s.t.} \quad \sum_{i=1}^{n+1} y_{ij} = \sum_{i=1}^{n+1} y_{ji} = 1 \qquad \forall j \in \{1,\ldots,n\} \tag{11}$$

$$\sum_{i=1}^{n} y_{n+1,i} = \sum_{i=1}^{n} y_{i,n+1} = m \tag{12}$$

$$\sum_{i,j\in S} y_{ij} \leq |S| - k(S) \qquad \forall S \subseteq \{1,\ldots,n\} \tag{13}$$

Here, Objective (10) minimizes the total travel distance, constraints (11) ensure each city is visited once, and constraints (12) ensure exactly m tours should be considered. Constraints (13) are subtour elimination constraints, where $k(S) = 1 + \lfloor |S|/(n/m) \rfloor$. We solve this formulation by adding the subtour elimination constraints dynamically throughout the branch-and-bound search, both for integer LP relaxations and for fractional LP relaxations.

7.2. Nearest Neighbor Heuristic

We generalize the Nearest Neighbor (NN) heuristic of Mahajan, Rao, and Peters (1998) to handle larger buffer sizes than 2. Firstly, we select the m request locations for which the distance to the I/O point is the smallest. These m requests are the first to be retrieved, i.e., serve as $r_1,\ldots,r_m$ in

the retrieval sequence. Then, for $i = m+1, ..., n-m$, the selection process finds the non-handled request that is located the closest to request r_i $(0 < i < n-m)$ and adds it as request r_{i+m} in the request sequence. If we continue from a previous set of requests (in a block sequencing approach), the first step that selects the m retrievals closest to the I/O-point is to be skipped. Due to the simple structure of this heuristic, based solely on the proximity of storage locations, the resulting retrieval sequence is the same for both objectives, i.e., the makespan with waiting times and the makespan without waiting times.

7.3. BW Heuristic

For the case of minimizing makespan with waiting times, Bozer and White (1996) present a heuristic procedure called Heuristic A in the original paper. However, for clarity, we will refer to it by the first letters of the paper's authors (BW). This heuristic was designed for a buffer size of 2 and is presented here with a minor adaptation to make it functional for any buffer size. Define $t_{(i)j}$ as the time needed by the S/R to store load r_i and retrieve load j, define $p_{(i)}$ as the time the worker requires to pick from the load of retrieval r_i, and define $C_{(i)j} = \max\{0, t_{(i)j} - p_{(i)}\}$ as a measure for worker idle time. The heuristic starts by randomly selecting m requests that serve as $r_1, \ldots, r_m$ in the retrieval sequence. Then, the remaining requests are selected by sequentially finding for $i = 1, ..., n-m$ the non-handled request j that minimizes $C_{(i)j}$ and adds j to the retrieval sequence as request r_{i+m}. As with the NN heuristic, in a block sequencing approach, the first step that selects the first m retrievals is to be skipped.

7.4. General Variable Neighborhood Search Heuristic

As the mTSPEV is a specific case of the mTSP, we adapt a recent heuristic for the mTSP to compare the performance of our simulated annealing heuristic with the latest mTSP heuristics. To this end, we modify the General Variable Neighborhood Search (GVNS) heuristic introduced in Soylu (2015). Mainly, GVNS heuristics are built upon the idea that while it might not be possible to find a better solution in one neighborhood of the current solution, there might be another neighborhood structure in which a better solution can be found. Different neighborhoods of the current solution can be obtained by applying different operators (simple procedures that create new solutions). Altogether, at a current solution, the GVNS heuristic searches for a better solution in one neighborhood. If no better solution is found, another neighborhood structure is investigated, and so on until no improvement can be found in any specified neighborhood structure.

To generate new candidate solutions, the GVNS heuristic of Soylu (2015) uses six neighborhood structures that are valid for the mTSP. However, only two of these structures are valid for the mTSPEV. The two valid neighborhoods are those obtained by applying either an inter-tour city swap or a intra-tour city swap, referred to as *two-point move* and *2-opt move*, respectively. The

other four neighborhood structures proposed in Soylu (2015) are invalid for the mTSPEV as they do not preserve an equal number of visits per salesman, a necessity in the mTSPEV. With regards to the objective function, the heuristic can minimize both the makespan without waiting times (i.e., Equation 1) and the makespan with waiting times (Equation 9). For initializing the heuristic, we use the NN solution for faster convergence. As for the rest of the heuristic, we use the same heuristic as presented in Soylu (2015).

7.5. Simulated Annealing

Our Simulated Annealing (SA) heuristic uses permutations of the retrieved loads as the encoding scheme. Each encoded candidate solution represents the retrieval sequence $r = (r_1, \ldots, r_n)$, where r_1 is the first item to be retrieved and stored, and r_n is the last item to be retrieved and stored. The retrieval sequence may also be seen as the order in which the picker receives loads. Figure 3 presents the encoded solution of the example from Figure 2 which has $n = 8$ loads, buffer size of $m = 2$, and an S/R travel sequence of

$$(v_0, v_1^r, v_0)(v_0, v_5^r, v_0)(v_0, v_1^s, v_2^r, v_0)(v_0, v_5^s, v_6^r, v_0)(v_0, v_2^s, v_3^r, v_0)...$$
$$...(v_0, v_6^s, v_7^r, v_0)(v_0, v_3^s, v_4^r, v_0)(v_0, v_7^s, v_8^r, v_0)(v_0, v_4^s, v_0)(v_0, v_8^s, v_0).$$

Notice that the encoded solution corresponds to the retrieval sequence (v_i^r) in the S/R work sequence. As explained in Section 4, this S/R travel sequence corresponds to the mTSPEV sequences $(v_0, v_1, v_2, v_3, v_4, v_0)$ and $(v_0, v_5, v_6, v_7, v_8, v_0)$ and vice versa.

The SA heuristic generates new candidate solutions r by using standard pairwise exchanges between requests. Notice, however, that by performing pairwise exchanges to our encoded solution we either swap vertices between two mTSPEV routes or swap the order in which two vertices are visited within a salesman's route. The latter happens when the pairwise exchange happens between positions that are separated by exactly m positions, whereas the former occurs otherwise. For example, if in the encoded solution in Figure 3 one exchanges position 1 with position 3 (represented by loads 1 and 2, respectively), so that the candidate solution becomes (2,5,1,6,3,7,4,8), then the corresponding mTSPEV solution becomes $(v_0, v_2, v_1, v_3, v_4, v_0)$ and $(v_0, v_5, v_6, v_7, v_8, v_0)$. That is, the visiting sequence for salesman 1 changed. On the other hand, switching positions 1 and 2 (i.e., loads 1 and 5) in the original encoded solution creates the mTSPEV solution $(v_0, v_5, v_2, v_3, v_4, v_0)$ and $(v_0, v_1, v_6, v_7, v_8, v_0)$. That is, the salesmen swap cities.

With regards to the objective function, we can use either the makespan without waiting times (i.e., Equation 1) or the makespan with waiting times (Equation 9). The newly generated candidate solution r is automatically accepted as the incumbent solution if its makespan, $M(r)$, is smaller than the last registered makespan, $M(r^*)$. If this is not the case, the new solution is accepted with the Boltzmann acceptance probability $\exp[-(M(r) - M(r^*))/T(t)]$, where $T(t)$ is the temperature

1 5 2 6 3 7 4 8

Figure 3 Encoded solution for example in Figure 2.

annealing schedule in the t'th iteration of the SA heuristic. If the new solution is accepted, we set r to become the new r^*. In our heuristic, the initial temperature and annealing schedule are set to the sigmoid-like function $T(t) = n^5/(1+e^{50t/n^5-5})$, along with the termination criterion of n^5 candidate solutions considered. These SA parameters were methodically fine-tuned through preliminary experimentation. Although SA is designed to converge irrespective of its initial solution, we opted to use the NN solution as initialization for SA to improve its convergence speed. Concerning the remaining details of the SA, we apply the traditional SA scheme in Eglese (1990).

8. Results

In this section, we compare heuristic and optimal solutions in terms of their average, best-case, and worst-case performance and numerically study system characteristics. The heuristics have been described in Section 7. Optimal solutions are obtained using the branch-and-cut algorithm (see Section 7). Optimal solutions and LP relaxations are obtained using CPLEX 12.8 and its associated callbacks coded in C++, based on the library provided by Uit het Broek et al. (2020). The heuristics are programmed and executed in Java SE 7 on a group of computers with 2.30GHz processors and 4GB of RAM.

In these computations, we analyze the results of the various solution methods in a multitude of settings by varying the number of products to be retrieved (n) and the buffer size of the system (m). For n we consider the values 20, 30, and 40, whereas for m we consider 2, 3, 4, and 5. In line with Bozer and White (1996), we assume locations of requests in the rack to be distributed continuously and uniformly. Rack dimensions are set to 20×20 in most experiments. The effects of rack dimensions on performance are evaluated in Section 8.3.

8.1. Experiments without waiting times

Tables 2, 3, and 4 show the performance of the Nearest Neighbor (NN), General Variable Neighborhood Search (GVNS), and Simulated Annealing (SA) heuristics with the optimal solution, respectively. The SA and GVNS heuristics are repeated ten times on each instance to account for the procedures' uncertainty. All instances are solved to optimality using the branch-and-cut algorithm. The NN heuristic originates from Mahajan, Rao, and Peters (1998) with minor adaptations to make it functional for any buffer size. The GVNS heuristic is an adapted version of the mTSP heuristic proposed in Soylu (2015), while the SA heuristic is newly proposed for this application. The first two columns describe the number of products n and the buffer size m. For each such combination, we considered 500 instances for $n = 20$ and 100 instances for $n > 20$.

For the NN heuristic, depicted in Table 2, we provide its average difference to optimality ($\Delta^{\text{OPT}}_{\text{AVG}}$), its average solution time in seconds (time(s)), and the fraction of instances solved to optimality ($\#_{\text{opt}}$). For the optimal solution we provide its average time as well as the maximum time among the instances being solved (max time(s)). For the GVNS and SA heuristics, depicted in Tables 3 and 4, we provide the average over all instances of the best-case, mean, and worst-case solution found in the 10 runs of the heuristic. We depict these results in the columns $\Delta^{\text{OPT}}_{\text{BEST}}$, $\Delta^{\text{OPT}}_{\text{AVG}}$, and $\Delta^{\text{OPT}}_{\text{WORST}}$.

Table 2 Average percent difference between optimal solution and NN.

		NN			OPT	
n	m	$\Delta^{\text{OPT}}_{\text{AVG}}$ %	time (s)	$\#_{\text{OPT}}$	time (s)	max time(s)
20	2	22.41	0.00	0.00	29.05	228.80
	3	24.95	0.00	0.00	23.31	265.23
	4	25.30	0.00	0.00	23.07	218.10
	5	25.03	0.00	0.00	23.32	237.69
30	2	27.26	0.00	0.00	27.05	117.95
	3	28.68	0.00	0.00	30.53	121.04
	4	31.13	0.00	0.00	21.28	118.77
	5	28.66	0.00	0.00	33.15	187.93
40	2	29.93	0.00	0.00	77.99	305.42
	3	32.31	0.00	0.00	48.05	435.30
	4	33.21	0.00	0.00	257.08	10419.40
	5	29.91	0.00	0.00	247.08	10800.00*

Table 3 Average percent difference between optimal solution and GVNS.

		GVNS (10 replications)					OPT	
n	m	$\Delta^{\text{OPT}}_{\text{BEST}}$ %	$\Delta^{\text{OPT}}_{\text{WORST}}$ %	$\Delta^{\text{OPT}}_{\text{AVG}}$ %	Time (s)	$\#_{\text{OPT}}$	Time (s)	Max Time(s)
20	2	2.55	6.38	4.35	0.40	0.32	29.05	228.80
	3	2.15	6.83	4.28	0.40	0.23	23.31	265.23
	4	0.49	3.81	1.85	0.40	0.62	23.07	218.10
	5	0.19	2.45	1.07	0.40	0.77	23.32	237.69
30	2	3.65	8.16	5.91	3.21	0.12	27.05	117.95
	3	4.16	9.21	6.61	3.22	0.10	30.53	121.04
	4	3.84	9.67	6.67	3.22	0.00	21.28	118.77
	5	1.17	6.56	3.54	3.17	0.18	33.15	187.93
40	2	5.28	10.04	7.67	19.32	0.02	77.99	305.42
	3	6.96	11.99	9.43	19.17	0.01	48.05	435.30
	4	4.51	10.70	7.75	19.54	0.02	257.08	10419.40
	5	3.44	9.13	6.26	19.56	0.00	247.08	10800.00*

* one instance terminated at 10800 s before closing optimality gap.

Table 4 Average percent difference between optimal solution and SA.

		SA (10 replications)					OPT	
n	m	$\Delta^{\text{OPT}}_{\text{BEST}}$ %	$\Delta^{\text{OPT}}_{\text{WORST}}$ %	$\Delta^{\text{OPT}}_{\text{AVG}}$ %	Time (s)	$\#_{\text{OPT}}$	Time (s)	Max Time(s)
20	2	0.00	1.66	0.48	0.64	0.99	29.05	228.80
	3	0.02	1.71	0.58	0.64	0.97	23.31	265.23
	4	0.00	0.23	0.05	0.64	1.00	23.07	218.10
	5	0.00	0.04	0.01	0.64	1.00	23.32	237.69
30	2	0.07	2.92	1.06	5.68	0.83	27.05	117.95
	3	0.07	1.48	0.62	5.70	0.81	30.53	121.04
	4	0.99	2.93	1.80	5.73	0.07	21.28	118.77
	5	0.01	0.52	0.17	5.74	0.96	33.15	187.93
40	2	0.25	3.98	1.72	27.57	0.51	77.99	305.42
	3	0.62	3.07	1.56	27.51	0.21	48.05	435.30
	4	0.17	1.70	0.78	27.83	0.62	257.08	10419.40
	5	0.05	1.16	0.52	28.35	0.81	247.08	10800.00*

* one instance terminated at 10800 s before closing optimality gap.

From Table 2 we observe that the average percent difference for NN lies between 20 and 35 percent from optimality. Moreover, the performance of NN seems to deteriorate when n increases. Table 3 shows that the GVNS heuristic performs consistently better than NN, with an average optimality gap between 2 and 12 percent. The average best-case performance, i.e., the average of the minimum over ten runs for each instance, lies within 7% of optimality, with better performance for larger buffer sizes due to the more restricted solution space. The average worst-case performance of up to 12% also shows this heuristic may not be reliable enough for practical use.

On the other hand, the performance of the SA heuristic in Table 4 shows that the average percent difference between the SA and optimal solution is about 1.80% or less. Looking to the best-case performance, we find high-quality solutions with optimality gaps within 1%. The run times for the GVNS and SA heuristics are significantly lower than for the branch-and-cut algorithm, as can be seen from Tables 3 and 4. Average run times for the branch-and-cut algorithm exceed 4 minutes per instance for the larger instances, while this is at most 30 seconds for the heuristics. Moreover, as is common with branch-and-cut algorithms, also here we observe that run times for generating optimal solutions vary strongly. For some instances, finding an optimal solution required several hours. Considering solutions would need to be calculated in practice while the operation is ongoing, the shorter and more predictable run times of the heuristics appear preferable.

8.2. Experiments with waiting times

Next, we perform a number of experiments to analyze the effects of waiting times on makespan and performance of the heuristics. To benchmark, we include the BW heuristic from Bozer and White (1996), which was developed explicitly to mitigate waiting times. We conduct our comparisons in three different scenarios, namely the picking time is (1) low, (2) medium, or (3) high, when

compared to the expected dual-command cycle time of the S/R. In particular, for scenario (1): $p = 0.5 \cdot \mathbb{E}[DC]$, for scenario (2): $p = \mathbb{E}[DC]$, and for scenario (3): $p = 1.5 \cdot \mathbb{E}[DC]$, where $\mathbb{E}[DC]$ denotes the expected dual-command cycle time. Derivations for $\mathbb{E}[DC]$ can be found in Bozer and White (1984). The other settings are taken the same as in the previous experiments.

The results of all three scenarios are provided in Table 5. The columns present similar information as in Tables 2-4, but we now compare the fraction of times a particular heuristic outperforms the BW heuristic ($\#_{\text{BW}}$) instead comparing it with the branch-and-cut method. Note that inclusion of waiting times results in a non-linear optimization problem that our branch-and-cut method cannot solve. We omit the average best-case and worst-case performance of the SA and GVNS, because these statistics are similar to the average performance and do not provide additional information.

From this table, we observe that the average performance of NN, GVNS, and SA is considerably better than BW for relatively small pick times. However, whereas the GVNS and the SA always outperform the BW heuristic, the NN fails to do so even for the smallest instances with low picking times. Comparing GVNS and SA, we observe that again the SA heuristic finds better solutions than the GVNS heuristic. However, with waiting times the gaps are smaller than without waiting times.

Zooming in on the differences in performance for varying picking times, we observe a similar pattern across all heuristics. For increasing waiting times, the average percent differences between the heuristics have decreased, as well as its improvement over the BW heuristic. A likely explanation for this trend is that the relative contribution of travel time to the total makespan is diminishing, and hence request scheduling is less impactful. Or put reversely, the smaller S/R waiting times are, the more useful request sequencing is.

8.3. Experiments with varying rack dimensions

In all previous experiments, we assumed that the rack dimensions are 20×20. Here, we investigate the effects of varying rack dimensions by comparing dimensions of 40×10, 20×20, and 10×40. We take $n = 20$ for the number of requests and $p = 1$ for the worker's picking time. Table 6 presents the results by listing the average improvement over BW over 100 instances and the fraction of times a better solution is found, in terms of the makespan including waiting times, of NN, GVNS, and SA.

As before, SA continues to outperform NN, BW, and GVNS on the average performance. Furthermore, we recognize that both the GVNS and SA heuristics are not impacted by the change in rack dimensions, as the average differences with BW remain roughly the same. It should be noted this also holds for the average best and worst-case results, but they were omitted from the table as the values were very close to the average results. On the other hand, the rack dimensions have a significant impact on the performance of NN. Especially, the worst-case results deteriorate

Table 5 **Average percent difference in performance between BW and NN/GVNS/SA for low, medium, and high picking times.**

			NN			GVNS (10 rep.)			SA (10 rep.)		
n	m	p	$\Delta^{\text{BW}}_{\text{AVG}}$ %	Time (s)	$\#_{BW}$	$\Delta^{\text{BW}}_{\text{AVG}}$ %	Time (s)	$\#_{BW}$	$\Delta^{\text{BW}}_{\text{AVG}}$ %	Time (s)	$\#_{BW}$
20	2	0.5	-4.73	0.00	0.90	-8.26	8.29	1.00	-8.90	7.03	1.00
		1	-0.69	0.00	0.63	-5.61	8.61	1.00	-5.69	7.34	1.00
		1.5	-1.20	0.00	0.77	-2.95	8.56	1.00	-2.95	7.24	1.00
	3	0.5	-3.37	0.00	0.87	-7.48	8.40	1.00	-8.08	7.10	1.00
		1	0.87	0.00	0.40	-5.82	8.79	1.00	-5.83	7.50	1.00
		1.5	-1.47	0.00	0.90	-3.50	8.78	1.00	-3.50	7.45	1.00
	4	0.5	-2.30	0.00	0.80	-6.99	8.67	1.00	-7.37	7.37	1.00
		1	0.42	0.00	0.43	-6.09	8.82	1.00	-6.14	7.54	1.00
		1.5	-1.73	0.00	0.87	-3.87	9.04	1.00	-3.87	7.72	1.00
	5	0.5	-1.67	0.00	0.67	-6.42	8.92	1.00	-6.71	7.57	1.00
		1	-0.17	0.00	0.53	-5.65	8.86	1.00	-5.80	7.56	1.00
		1.5	-1.25	0.00	0.80	-3.45	9.24	1.00	-3.45	7.85	1.00
30	2	0.5	-7.28	0.00	1.00	-10.03	91.78	1.00	-10.43	101.79	1.00
		1	-1.53	0.00	0.87	-4.90	97.23	1.00	-4.92	108.72	1.00
		1.5	-1.49	0.00	0.93	-2.75	94.84	1.00	-2.75	105.24	1.00
	3	0.5	-5.40	0.00	0.97	-8.47	86.33	1.00	-9.10	95.62	1.00
		1	-0.32	0.00	0.50	-4.90	99.04	1.00	-4.90	110.83	1.00
		1.5	-1.25	0.00	0.90	-2.58	97.74	1.00	-2.58	108.58	1.00
	4	0.5	-4.56	0.00	0.93	-7.84	98.62	1.00	-8.51	109.05	1.00
		1	0.69	0.00	0.47	-4.41	100.81	1.00	-4.42	112.28	1.00
		1.5	-0.69	0.00	0.80	-2.24	102.43	1.00	-2.24	113.92	1.00
	5	0.5	-3.92	0.00	0.90	-8.27	100.54	1.00	-8.85	111.53	1.00
		1	1.09	0.00	0.30	-4.21	101.26	1.00	-4.23	112.53	1.00
		1.5	-0.64	0.00	0.83	-2.26	101.28	1.00	-2.26	112.16	1.00
40	2	0.5	-7.74	0.00	1.00	-10.13	796.82	1.00	-10.69	783.54	1.00
		1	-0.68	0.00	0.67	-3.45	775.92	1.00	-3.46	771.91	1.00
		1.5	-0.55	0.00	0.77	-1.62	770.88	1.00	-1.62	762.93	1.00
	3	0.5	-5.37	0.00	0.90	-9.19	808.08	1.00	-10.01	794.17	1.00
		1	-0.26	0.00	0.57	-3.80	834.13	1.00	-3.80	829.69	1.00
		1.5	-0.67	0.00	0.80	-1.73	778.24	1.00	-1.73	771.22	1.00
	4	0.5	-6.79	0.00	1.00	-10.46	830.84	1.00	-11.08	816.18	1.00
		1	0.39	0.00	0.47	-3.97	823.97	1.00	-3.97	811.69	1.00
		1.5	-0.75	0.00	0.90	-1.90	799.29	1.00	-1.90	792.33	1.00
	5	0.5	-6.24	0.00	1.00	-9.36	828.29	1.00	-10.18	813.43	1.00
		1	0.64	0.00	0.43	-3.43	825.19	1.00	-3.43	809.93	1.00
		1.5	-0.47	0.00	0.77	-1.73	807.88	1.00	-1.73	797.74	1.00
Aggregated Averages											
	$p = 0.5$		-4.95	0.00	0.91	-8.58	306.30	1.00	-9.16	304.53	1.00
	$p = 1$		0.04	0.00	0.52	-4.69	307.72	1.00	-4.72	308.12	1.00
	$p = 1.5$		-1.01	0.00	0.84	-2.55	299.02	1.00	-2.55	299.53	1.00
	$m = 2$		-2.88	0.00	0.84	-5.52	294.77	1.00	-5.71	295.08	1.00
	$m = 3$		-1.92	0.00	0.76	-5.28	303.28	1.00	-5.50	303.57	1.00
	$m = 4$		-1.70	0.00	0.74	-5.31	309.17	1.00	-5.50	308.67	1.00
	$m = 5$		-1.40	0.00	0.69	-4.97	310.16	1.00	-5.18	308.92	1.00
	$n = 20$		-1.44	0.00	0.71	-5.51	8.75	1.00	-5.69	7.44	1.00
	$n = 30$		-2.11	0.00	0.78	-5.24	97.66	1.00	-5.43	108.52	1.00
	$n = 40$		-2.38	0.00	0.77	-5.06	806.63	1.00	-5.30	796.23	1.00

Table 6 **Average percent difference in performance (100 instances) between BW and NN/GVNS/SA for varying rack dimensions with n=20 and p=1.**

		NN			GVNS (10 rep.)			SA (10 rep.)		
m	Dimension	$\Delta^{\text{BW}}_{\text{AVG}}$ %	Time (s)	$\#_{BW}$	$\Delta^{\text{BW}}_{\text{AVG}}$ %	Time (s)	$\#_{BW}$	$\Delta^{\text{BW}}_{\text{AVG}}$ %	Time (s)	$\#_{BW}$
2	10×40	-0.52	0.00	0.61	-5.05	8.19	1.00	-5.13	7.03	1.00
	20×20	-1.46	0.00	0.88	-3.30	8.21	1.00	-3.30	6.97	1.00
	40×10	-1.38	0.00	0.69	-5.80	8.17	1.00	-5.87	7.03	1.00
3	10×40	5.40	0.00	0.21	-8.52	8.48	1.00	-8.56	7.28	1.00
	20×20	-1.80	0.00	0.92	-3.89	8.38	1.00	-3.89	7.12	1.00
	40×10	6.83	0.00	0.02	-8.61	8.48	1.00	-8.69	7.27	1.00
4	10×40	5.30	0.00	0.19	-8.34	8.75	1.00	-8.41	7.53	1.00
	20×20	-1.53	0.00	0.83	-3.62	8.75	1.00	-3.62	7.47	1.00
	40×10	7.31	0.00	0.05	-7.51	8.72	1.00	-7.57	7.51	1.00
5	10×40	7.09	0.00	0.14	-6.76	8.96	1.00	-6.84	7.66	1.00
	20×20	-1.13	0.00	0.75	-3.31	8.98	1.00	-3.31	7.66	1.00
	40×10	6.95	0.00	0.11	-6.40	8.91	1.00	-6.55	7.65	1.00

considerably if the dimension becomes either 40×10 or 10×40. This could be caused by the fact that the maximum possible distance increases as the rack becomes either wider or taller, which in turn could lead to much larger errors as NN schedules the final requests that can be far away from the I/O point.

8.4. Experiments on block sequencing

Finally, we analyze the performance of NN, BW, GVNS, and SA under a block sequencing approach, as explained in Section 3. For this experiment we consider a setting in which a set of 120 requests in a 20×20 rack needs to be handled, while starting and ending with an empty buffer of size 2. First, the heuristics are run on the full set of 120 requests to determine their performance in a setting of full information. Next, we consider several settings of the block sequencing approach: 2 batches of size 60, 3 batches of size 40, 4 batches of size 30, 6 batches of size 20, and 12 batches of size 10. For every setting, we separately run our heuristics on each of the batches, and then compute the total makespan including waiting times if all batches were handled sequentially. This yields the actual makespan for all 120 loads when we apply the block sequencing approach. The results of our experiments are presented in Figure 4.

In Figure 4, the performance of each heuristic is normalized with respect to the case where SA is given full information, as it yields the smallest makespan. In other words, the curves show each heuristic's makespan divided by the makespan of the SA solution at a batch size of 120. As seen before, we observe that SA also performs best under a block sequencing approach. It is interesting to notice that smaller batch sizes yield efficiency losses of only a few percent in the cases studied here. For batch sizes larger than 20, SA under a block sequencing approach achieves makespans that are within 1% of its makespan under full information of all 120 requests. In addition, we note

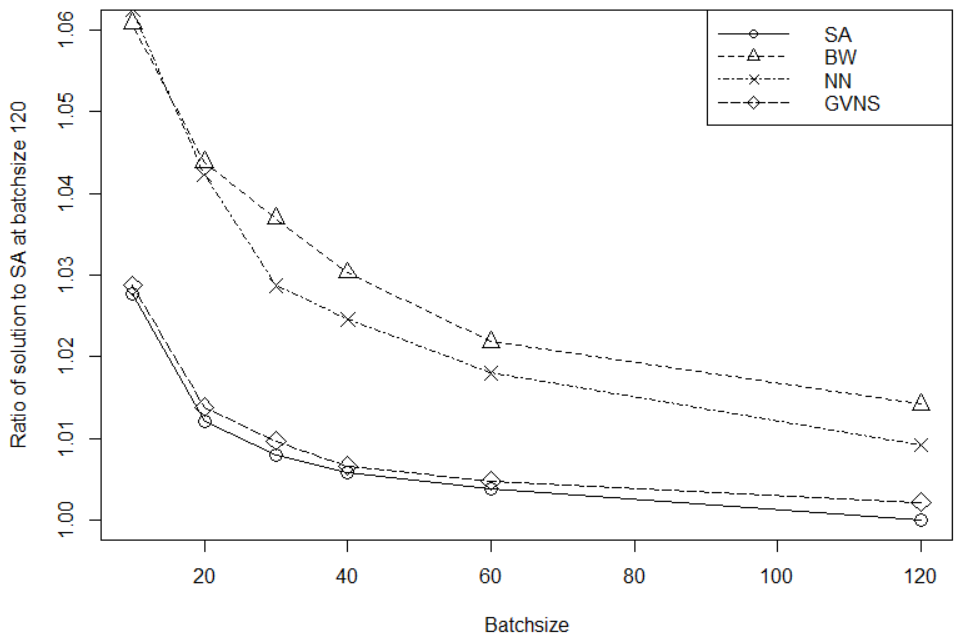

Figure 4 **Comparison of the performance of NN, BW, GVNS, and SA under a block sequencing approach (Batchsizes: 10, 20, 30, 40, 60) with the performance of SA under full information (Batchsize: 120).**

that the total running time of the SA for four batches of size 30 is only 375 seconds, whereas computation for the full batch of size 120 takes on average 9218 seconds. Hence, by running SA on smaller batches, which is computationally favorable, the efficiency losses appear limited.

9. Conclusions

In this paper, we studied an end-of-aisle automated storage/retrieval system (AS/RS). The problem under consideration consists of scheduling a given list of storage and retrieval requests for this system; this is based on a scenario faced by warehouses that retrieve loads from an automated warehouse to fulfill a list of orders within a given time period. After a load is retrieved, a worker gathers products from the load, before requesting the AS/RS to return the load to storage. The objective is to minimize the total makespan for serving all requests.

The problem is shown to be mathematically equivalent to a problem we termed the multiple Traveling Salesmen Problem with an equal number of visits (mTSPEV) if the S/R incurs no waiting times. The mTSPEV is a constrained version of the multiple Traveling Salesmen Problem (mTSP) where the salesmen must visit an equal number of cities. It is argued that the mTSPEV is an $\mathcal{NP}$-hard problem as it may be seen as a generalization of the Traveling Salesman Problem (TSP). Several interesting structural properties for the mTSPEV are presented, which show that

the problem is unusual. For example, it is shown that unlike in the general mTSP, the length of the optimal route for the mTSPEV may actually strictly decrease when the number of cities is increased.

This paper also presents a branch-and-cut method and several heuristics for the problem. Experimental results suggest that the proposed simulated annealing heuristic consistently outperforms the nearest neighbor heuristic by 20% as well as the generalized variable neighborhood search heuristic of Soylu (2015) by more than 2%, on instances without waiting times. On average, the simulated annealing heuristic performs within 1.8% from the optimal solution on instances without waiting times. On instances with waiting times, our heuristic is around 5% better than the nearest neighbor and Bozer and White (1996) heuristics, while being slightly better than the generalized variable neighborhood search heuristic of Soylu (2015).

In practice, an AS/RS typically exhibits multiple storage and retrieval (S/R) machines that work independently in aisles. Whereas the focus of this paper is on the fundamental understanding of a single end-of-aisle AS/RS, a customer order may require products from multiple aisles in practice. Several options exist in practice to connect multiple aisles and pick stations. Particularly in the current age of fast response times and high customer service, it might be of interest to study this scheduling process jointly with the sequencing of each S/R machine. We especially advocate fast heuristic solutions for such systems, as this will consist of 10,000's or more loads to be scheduled. Decomposition approaches, in which single end-of-aisle AS/RS request sequencing problems are identified, can however, still benefit from the approaches presented in this paper.

Another avenue for further research is to study linear approximations of the end-of-aisle AS/RS request sequencing problem with waiting times. Especially approximation guarantees or lowerbound approximations would be useful, as it improves assessment of heuristics' solution quality.

Acknowledgments

This work was partially supported by TKI Dinalog, the Dutch Institute for Advanced Logistics.

Appendix A: Proof of Theorem 1

For the first part of this proof, we will lift the requirement that loads leave the buffer based on the First-In-First-Out rule. That is, loads can be retrieved in any sequence once they are in the buffer. We will show that the end-of-aisle AS/RS request sequencing problem then is equivalent to the regular mTSP.

Take the end-of-aisle AS/RS request sequencing problem. We transform this arc routing problem into a vertex routing problem. We do this by replacing the required arcs by vertices and adjusting the distances accordingly.

$\overline{v}_i^r$ is the node that replaces the arc (v_i^r, v_0)

$\overline{v}_i^s$ is the node that replaces the arc (v_0, v_i^s)

We denote the length of an arc (x, y) in the original arc routing problem as $d(x, y)$, which simply equals the physical distance between the vertices x and y. In transforming the arc routing problem into a vertex routing problem, the distances must be adjusted. Since required arcs must be traversed anyway, their total length simply forms a constant in the problem's objective function, so we can ignore their length in the new formulation. However, some of the other distances are impacted. Our new distances are as follows, for any indices $i, j \in \{1, 2, ..., n\}$.

$\overline{d}(\overline{v}_i^s, \overline{v}_j^s) = d(v_i^s, v_0)$

$\overline{d}(\overline{v}_i^r, \overline{v}_j^s) = 0$

$\overline{d}(\overline{v}_i^s, \overline{v}_j^r) = d(v_i^s, v_j^r)$

$\overline{d}(\overline{v}_i^r, \overline{v}_j^r) = d(v_0, v_j^r)$

$\overline{d}(v_0, \overline{v}_j^s) = 0$

$\overline{d}(\overline{v}_i^s, v_0) = d(v_i^s, v_0)$

$\overline{d}(v_0, \overline{v}_j^r) = d(v_0, v_j^r)$

$\overline{d}(\overline{v}_i^r, v_0) = 0$

$\overline{d}(v_0, v_0) = 0.$

These distances have the following four properties:

(1) $\overline{d}(\overline{v}_j^r, \overline{v}_i^r) = \overline{d}(\overline{v}_k^r, \overline{v}_i^r)$ for all $i \neq j \neq k$

(2) $\overline{d}(\overline{v}_i^r, x) = \overline{d}(\overline{v}_j^r, x)$ for all $i \neq j$ and for all vertices x

(3) $\overline{d}(\overline{v}_i^s, \overline{v}_j^s) = \overline{d}(\overline{v}_i^s, \overline{v}_k^s)$ for all $i \neq j \neq k$

(4) $\overline{d}(x, \overline{v}_i^s) = \overline{d}(x, \overline{v}_j^s)$ for all $i \neq j$ and for all vertices x

In our new formulation, we no longer need to explicitly add the I/O-point in the route tuple for every storage and retrieval request, since all distances are accounted for by the new distance definition. Therefore vertex v_0 will only appear twice, at the beginning and at the end. Due to properties (1) and (2), for a given route, we can remove any v_i^r that follows a retrieval request v_j^r and put it after any other retrieval v_k^r at no additional cost. For example, the routes $(v_0, v_1^r, v_2^r, v_1^s, v_2^s, v_3^r, v_4^r, v_3^s, v_4^s, v_0)$ and $(v_0, v_1^r, v_4^r, v_2^r, v_1^s, v_2^s, v_3^r, v_3^s, v_4^s, v_0)$ have equal length. This will also maintain the requirement that loads must be picked up before they can be stored. Similarly, due to properties (3) and (4) a storage request v_i^s that is followed by another storage request v_j^s can be moved at no cost to precede any other storage request v_k^s. We use this property to move appropriate retrievals to the front, and appropriate storages to the end of the route as much as possible. Thus

our example turns to $(v_0, v_1^r, v_4^r, v_2^r, v_2^s, v_3^r, v_3^s, v_1^s, v_4^s, v_0)$. Doing so consistently with all concerned occurrences, results in a route tuple with the following structure:

$(v_0, v_{(1)}^r, v_{(2)}^r, ..., v_{(m)}^r)(v_{(1)}^s, v_{(m+1)}^r)(v_{(2)}^s, v_{(m+2)}^r).....(v_{(n-m)}^s, v_{(n)}^r)(v_{(n-m+1)}^s, ..., v_{(n-1)}^s, v_{(n)}^s, v_0)$.

That is, a series of only retrievals, followed by a perfectly alternating sequence of storages and retrievals, and finally a series of only storages. With $v_{(i)}^r$ denoting the first retrieval that occurs in the route. We have added additional brackets at places in the tuple to indicate where the distance between two adjacent nodes is zero. We add occurrences of v_0 at no cost between each pair of adjacent storages and each pair of adjacent retrievals. For our example:

$(v_0, v_1^r, v_4^r, v_2^r)(v_2^s, v_3^r)(v_3^s, v_1^s, v_4^s, v_0)$ then turns into $(v_0, v_1^r)(v_0, v_4^r)(v_0, v_2^r)(v_2^s, v_3^r)(v_3^s, v_0)(v_1^s, v_0)(v_4^s, v_0)$

Next we sort the tuple at no cost like in the game of dominoes, so for example after (v_7^s, v_9^r) we put (v_9^s, v_{12}^r). This results in m sequences starting and ending in v_0 In the example:

$(v_0, v_2^r)(v_2^s, v_3^r)(v_3^s, v_0)$ and $(v_0, v_1^r)(v_1^s, v_0)$ and $(v_0, v_4^r)(v_4^s, v_0)$

It is fairly straightforward to see that this provides the routes for a solution of an mTSP problem, by leaving the distances unchanged and merging v_i^r and v_i^s to a single node for each $i = 1, ..., n$. Such merging of nodes does not impact the objective function value for any solution, since the distance between each pair of nodes is zero and each pair is always visited immediately one after the other.

This shows that any solution to the end-of-aisle AS/RS request sequencing problem has a corresponding solution in an mTSP with the same nodes (we now merged nodes back that we had split earlier) and the same distance matrix (in the solutions, we only have distances that remained unchanged in the redefinition of distances). This holds under the assumption that requests can be removed from the buffer in any sequence. To show the reverse, i.e., that any solution to an mTSP problem has a corresponding solution in an end-of-aisle AS/RS request sequencing problem with the same nodes and the same distance matrix, a similar, yet reversed, reasoning can be made.

Finally, we observe that we now have two problem formulations with a one-to-one correspondence in their solutions. If we impose any additional constraints on two equivalent problems, the resulting problems must also be equivalent. What remains to be proven therefore is that imposing equal visits to an mTSP problem is equivalent to imposing First-In-First-Out sequencing for a buffer of fixed size in the request sequencing problem. We start with a feasible solution to the mTSPEV, i.e., a solution to the mTSP with the additional restriction that all salesmen visit an equal number of cities. We write the solution in a tuple as:

$(v_0, v_{1,(1)}, v_{1,(2)}, ..., v_{1,(n/m)}, v_0)(v_0, v_{2,(1)}, v_{2,(2)}, ..., v_{2,(n/m)}, v_0)...(v_0, v_{m,(1)}, v_{m,(2)}, ..., v_{m,(n/m)}, v_0)$

where $v_{k,(i)}$ denotes the i^{th} city visited by salesman k in the particular solution. We split vertices $v_{k,(i)}$ into $v_{k,(i)}^s$ and $v_{k,(i)}^r$, where $v_{k,(i)}^r$ represents the salesman's arrival at city (i) and $v_{k,(i)}^s$ the salesman's departure from city (i) This gives:

$(v_0, v_{1,(1)}^r, v_{1,(1)}^s, v_{1,(2)}^r, v_{1,(2)}^s, ..., v_{1,(n/m)}^r, v_{1,(n/m)}^s, v_0)...$

$...(v_0, v_{m,(1)}^r, v_{m,(1)}^s, v_{m,(2)}^r, v_{m,(2)}^s, ..., v_{m,(n/m)}^r, v_{m,(n/m)}^s, v_0)$.

We observe that for the mTSPEV there is no requirement concerning timing of visiting cities across salesmen. So, we can assume that while salesman k is at a given city, all others must move to their next city, before salesman k is again allowed to move (i.e., they move in turns). Rewriting our tuple to reflect this timing, gives:

$(v_0, v^r_{1,(1)}, v^r_{2,(1)}, ..., v^r_{m,(1)})(v^s_{1,(1)}, v^r_{1,(2)})(v^s_{2,(1)}, v^r_{2,(2)})...(v^s_{m,(1)}, v^r_{m,(2)})...$
$..., (v^s_{1,(n/m-1)}, v^r_{1,(n/m)})(v^s_{2,(n/m-1)}, v^r_{2,(n/m)})...(v^s_{m,(n/m-1)}, v^r_{m,(n/m)})(v^s_{1,(n/m)}, ..., v^s_{m,(n/m)}, v_0).$

Now observe that in this tuple any $v^s_{k,(i)}$follows the corresponding $v^r_{k,(i)}$ exactly after m positions in the tuple. It can be verified that the above sequence does provide a feasible sequence for the S/R in the end-of-aisle AS/RS sequencing problem, at a length that differs only by the summed length of all required arcs (which is a constant) from the length of the solution we started with in the mTSPEV. This shows that m salesman taking turns, is equivalent to doing $m-1$ other jobs between the retrieval and storage of any load in the end-of-aisle system, which is the same as that there is a buffer of size m with a First-In-First-Out rule.

Appendix B: Proof of Property 1

First, assume $L^{TSP}_m(I(V)) < L^{TSPEV}_m(I(V))$, and consider the vertex set $U = \{v_{n+1}, \ldots, v_{nm}\}$ with all vertices of U located at the origin. Then, trivially, the mTSPEV route on $V \cup U$ will consist of one route following the TSP route on V and $m-1$ routes on U, in which the latter routes all have length zero. This implies:

$$L^{TSPEV}_m(I(V \cup U)) = L^{TSP}(I(V \cup U)) = L^{TSP}(I(V)) \leq L^{TSP}_m(I(V)) < L^{TSPEV}_m(I(V)).$$

To prove the reverse, assume $\exists U$ with $L^{TSPEV}_m(I(V \cup U)) < L^{TSPEV}_m(I(V))$. Then, since the solution space of the mTSPEV is a subset of the solution space of the mTSP,

$$L^{TSPEV}_m(I(V)) > L^{TSPEV}_m(I(V \cup U)) \geq L^{TSP}_m(I(V \cup U)) \geq L^{TSP}_m(I(V)).$$

Appendix C: Proof of Property 2

First property: Let $t_k = (v_0, v^k_{(1)}, v^k_{(2)}, ..., v^k_{(w(k))}, v_0)$ denote the route taken by salesman k in an optimal solution, where $w(k)$ denotes the number of cities in the route for salesman k, v_0 denotes the depot, and $v^k_{(i)}$ denotes the i^{th} vertex visited by salesman k. Denote by $t_{k\cup\ell}$ the simple concatenation of the routes of salesmen k and ℓ, omitting the middle visit to the depot. So, $t_{k\cup\ell} = (v_0, v^k_{(1)}, v^k_{(2)}, ..., v^k_{(w(k))}, v^\ell_{(1)}, v^\ell_{(2)}, ..., v^\ell_{(w(\ell))}, v_0)$. Then since the triangle inequality implies $d(v^k_{(w(k))}, v^\ell_{(1)}) \leq d(v^k_{(w(k))}, v_0) + d(v_0, v^\ell_{(1)})$ it follows that the length of $t_{k\cup\ell}$ is no longer than the sum of the lengths of t_k and t_ℓ. Using this fact for any two salesmen's routes, will suffice to reduce an (m+1)TSP to an mTSP with the stated property.

Second property: Consider the graph $G = (V, A)$ with $|V| = n = 6$, shown in Figure 5 (each point indicates two vertices located at the exact same location), and let $m = 2$. The optimal solution for $m = 2$ is given by $(v_0, v_1, v_2, v_3, v_0)(v_0, v_4, v_5, v_6, v_0)$ and its length is,

$$L^{TSPEV}_2(I(V)) = 8 + 4\sqrt{2}$$

The optimal solution for $m = 3$ is given by $(v_0, v_1, v_2, v_0)(v_0, v_3, v_4, v_0)(v_0, v_5, v_6, v_0)$ and its length is,

$$L^{TSPEV}_3(I(V)) = 12$$

Hence, this demonstrates $L^{TSPEV}_2(I(V)) = 8 + 4\sqrt{2} > 12 = L^{TSPEV}_3(I(V))$.

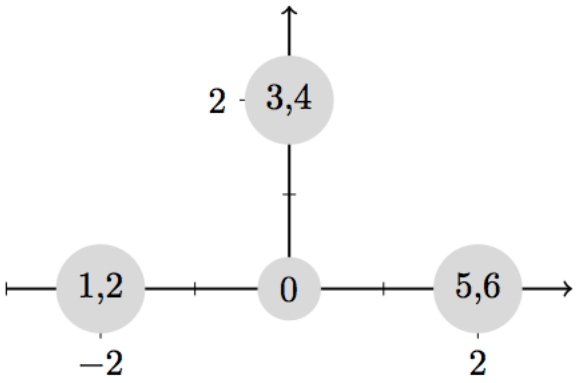

Figure 5 **Graph used in the proof of property 2.**

Appendix D: Proof of Property 3

Let $t_k = (v_0, v^k_{(1)}, v^k_{(2)}, \ldots, v^k_{(n/m)}, v_{(0)})$ denote the route taken by salesman k in an optimal solution, where v_0 denotes the depot, and $v^k_{(i)}$ denotes the i^{th} vertex visited by salesman k, and n/m denotes the number of cities visited by salesman k. Denote by $t_{k\cup\ell}$ the simple concatenation of the routes of salesmen k and ℓ, omitting the middle visit to the depot. So, $t_{k\cup\ell} = (v_0, v^k_{(1)}, v^k_{(2)}, \ldots, v^k_{(n/m)}, v^\ell_{(1)}, v^\ell_{(2)}, \ldots, v^\ell_{(n/m)}, v_0)$.

Then since the strict triangle inequality implies $d(v^k_{(n/m)}, v^\ell_{(1)}) < d(v^k_{(n/m)}, v_0) + d(v_0, v^\ell_{(1)})$ − provided that the vertices do not have identical coordinates − it follows that the length of $t_{k\cup\ell}$ is shorter than the sum of the lengths of t_k and t_ℓ. A (not necessarily optimal) solution to the mTSPEV for $I(V)$ with a salesmen can be found by repeated concatenation of routes found in an optimal solution to the mTSPEV for $I(V)$ with ba salesmen. An optimal solution to mTSPEV for $I(V)$ with a salesmen will be no longer than the solution obtained by concatenation. Hence, the property follows.

References

Ascheuer N, Grötschel M, Abdel-Hamid AAA, 1999 *Order picking in an automatic warehouse: Solving online asymmetric TSPs. Mathematical Methods of Operations Research* 49(3):501–515.

Azadeh K, De Koster R, Roy D, 2019 *Robotized and automated warehouse systems: Review and recent developments. Transportation Science* 53(4):917–945.

Bektas T, 2006 *The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega: The International Journal of Management Science* 34(3):209–219.

Bellmore M, Hong S, 1974 *Transformation of multisalesmen problem to the standard traveling salesman problem. Journal of the Association for Computing Machinery* 21(3):500–504.

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.

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.

Bozer YA, Aldarondo FJ, 2018 *A simulation-based comparison of two goods-to-person order picking systems in an online retail setting. International Journal of Production Research* 56(11):3838–3858.

Bozer YA, White JA, 1984 *Travel-time models for automated storage/retrieval systems. IIE Transactions* 16(4):329–338.

Bozer YA, White JA, 1990 *Design and performance models for end-of-aisle order picking systems. Management Science* 36(7):852–866.

Bozer YA, White JA, 1996 *A generalized design and performance analysis model for end-of-aisle order-picking systems. IIE Transactions* 28(4):271–280.

Cardona LF, Gue KR, 2020 *Layouts of unit-load warehouses with multiple slot heights. Transportation Science* 54(5):1332–1350.

Carlo HJ, Giraldo GE, 2012 *Toward perpetually organized unit-load warehouses. Computers & Industrial Engineering* 64(4):1003–1012.

Carter AE, Ragsdale CT, 2006 *A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research* 175(1):246–257.

Chen L, Langevin A, Riopel D, 2011 *A tabu search algorithm for the relocation problem in a warehousing system. International Journal of Production Economics* 129(1):147–156.

Dooly DR, Lee HF, 2008 *A shift-based sequencing method for twin-shuttle automated storage and retrieval systems. IIE Transactions* 40(6):586–594.

Eglese RW, 1990 *Simulated annealing: A tool for operational research. European Journal of Operational Research* 46(3):271–281.

Foley RD, Hackman ST, Park BC, 2004 *Back-of-the-envelope miniload throughput bounds and approximations. IIE Transactions* 36(3):279–285.

Frederickson G, Hecht M, Kim C, 1978 *Approximation algorithms for some routing problems. SIAM Journal on Computing* 7:178–193.

Gagliardi JP, Renaud J, Ruiz A, 2012 *Models for automated storage and retrieval systems: a literature review. International Journal of Production Research* 50(24):7110–7125.

Galle V, Barnhart C, Jaillet P, 2018 *Yard crane scheduling for container storage, retrieval, and relocation. European Journal of Operational Research* 271(1):288–316.

Gavish B, Srikanth K, 1986 *An optimal solution method for large-scale multiple traveling salesmen problems. Operations Research* 35(4):698–717.

Ghafurian S, Javadian N, 2011 *An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. Applied Soft Computing* 11(1):1256–1262.

Gharehgozli AH, 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.

Golden BL, Laporte G, Taillard ED, 1997 *An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. Computers & Operations Research* 24(5):445–452.

Gromicho J, Paixão J, Branco I, 1992 *Exact solution of multiple traveling salesman problems.* Akgül M, Hamacher H, Tüfekçi S, eds., *Combinatorial optimization*, 291–292, NATO ASI Series (Springer, Berlin).

Gu J, Goetschalckx M, McGinnis LF, 2007 *Research on warehouse operation: a comprehensive review. European Journal of Operational Research* 177(1):1–21.

Guardian, 2013 *Marks & spencer opens automated warehouse for online sales.* URL `http://www.guardian.co.uk/business/2013/may/08/marks-spencer-online-warehouse`.

Gue KR, Furmans K, Seibold Z, Uludag O, 2014 *Gridstore: A puzzle-based storage system with decentralized control. IEEE Trans. Automation Science and Engineering* 11(2):429–438.

Han MH, McGinnis LF, Shieh JS, White JA, 1987 *On sequencing retrievals in an automated storage/retrieval system. IIE Transactions* 19(1):56–66.

Hsu CY, Tsai MH, Chen WM, 1991 *A study of feature-mapped approach to the multiple travelling salesmen problem. IEEE international symposium on circuits and systems, 1991*, 1589–1592.

Jaghbeer Y, Hanson R, Johansson MI, 2020 *Automated order picking systems and the links between design and performance: a systematic literature review. International Journal of Production Research* 58(15):4489–4505.

Johnson DS, McGeoch LA, 1997 *The traveling salesman problem: A case study in local optimization.* Aarts E, Lenstra J, eds., *Local Search in Combinatorial Optimization*, 215–310 (John Wiley & Sons, London).

Kara I, Bektas T, 2006 *Integer linear programming formulations of multiple salesman problems and its variations. European Journal of Operational Research* 174(3):1449–1458.

Laporte G, 1992 *The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operations Research* 59(2):231–247.

Laporte G, Norbert Y, 1980 *A cutting plane algorithm for the m-salesmen problem. Journal of the Operational Research Society* 31(11):1017–1023.

Larki H, Yousefikhoshbakht M, 2014 *Solving the multiple traveling salesman problem by a novel meta-heuristic algorithm. Journal of Optimization in Industrial Engineering* 16:55–63.

Lee HF, Schaefer SK, 1997 *Sequencing methods for automated storage and retrieval systems with dedicated storage. Computers & Industrial Engineering* 32(2):351–362.

Liu W, Li S, Zhao F, Zheng A, 2009 *An ant colony optimization algorithm for the multiple traveling salesmen problem. 4th IEEE conference on industrial electronics and applications, 2009*, 1533–1537.

Mahajan S, Rao BV, Peters BA, 1998 *A retrieval sequencing heuristic for miniload end-of-aisle automated storage/retrieval systems. International Journal of Production Research* 36(6):1715–1731.

Man X, Zheng F, Chu F, Liu M, Xu Y, 2021 *Bi-objective optimization for a two-depot automated storage/retrieval system. Annals of Operations Research* 296:243–262.

Mirzaei M, De Koster RBM, Zaerpour N, 2017 *Modelling load retrievals in puzzle-based storage systems. International Journal of Production Research* 55(21):6423–6435.

Modares A, Somhom S, Enkawa T, 1999 *A self organizing neural network approach for multiple traveling salesman and vehicle routing problems. International Transactions in Operational Research* 6(6):591–606.

Nia AR, Haleh H, Saghaei A, 2017 *Dual command cycle dynamic sequencing method to consider ghg efficiency in unit-load multiple-rack automated storage and retrieval systems. Computers & Industrial Engineering* 111:89–108.

Park BC, Foley RD, Frazelle EH, 2006 *Performance of miniload systems with two-class storage. European Journal of Operational Research* 170(1):144–155.

Park BC, Foley RD, White JA, Frazelle EH, 2003 *Dual command travel times and miniload system throughput with turnover-based storage. IIE Transactions* 35(4):343–355.

Park BC, Frazelle EH, White JA, 1999 *Buffer sizing models for end-of-aisle order picking systems. IIE Transactions* 31(1):31–38.

Popović D, Vidović M, Bjelić N, 2014 *Application of genetic algorithms for sequencing of as/rs with a triple-shuttle module in class-based storage. Flexible Services and Manufacturing Journal* 26(3):432–453.

Potvin JY, Lapalme G, Rousseau JM, 1989 *A generalized k-opt exchange procedure for the mtsp. INFOR* 27(4):474–481.

Roodbergen KJ, Vis IFA, 2009 *A survey of literature on automated storage and retrieval systems. European Journal of Operational Research* 194(2):343–362.

Rostami AS, Mohanna F, Keshavarz H, Hosseinabadi AAR, 2015 *Solving multiple traveling salesman problem using the gravitational emulation local search algorithm. Applied Mathematics & Information Sciences* 9(2):699–709.

Russell RA, 1977 *Technical note - an effective heuristic for the m-tour traveling salesman problem with some side conditions. Operations Research* 25(2):517–524.

Sarin SC, Sherali HD, Judd JD, Tsai PFJ, 2014 *Multiple asymmetric traveling salesmen problem with and without precedence constraints: Performance comparison of alternative formulations. Computers & Operations Research* 51:64–89.

Sedighpour M, Yousefikhoshbakht M, 2012 *An effective genetic algorithm for solving the multiple traveling salesman problem. Journal of Optimization in Industrial Engineering* 8(1):73–79.

Singh A, Baghel AS, 2009 *A new grouping genetic algorithm approach to the multiple traveling salesperson problem. Soft Computing* 13(1):95–101.

Somhom S, Modares A, Enkawa T, 1999 *Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Computers & Operations Research* 26(4):395–407.

Song CH, Lee K, Lee WD, 2003 *Extended simulated annealing for augmented tsp and multi-salesmen tsp. Proceedings of the international joint conference on neural networks, 2003*, volume 3, 2340–2343.

Soylu B, 2015 *A general variable neighborhood search heuristic for multiple traveling salesmen problem. Computers & Industrial Engineering* 90:390–401.

Svestka JA, Huckfeldt VE, 1973 *Computational experience with an m-salesman traveling salesman algorithm. Management Science* 19(7):790–799.

Tanaka S, Araki M, 2009 *Routing problem under the shared storage policy for unit-load automated storage and retrieval systems with separate input and output points. International Journal of Production Research* 47(9):2391–2408.

Tang L, Liu J, Rong A, Yang Z, 2000 *A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. European Journal of Operational Research* 124(2):267–282.

Tompkins JA, White JA, Bozer YA, Tanchoco JMA, 2010 *Facilities Planning* (John Wiley & Sons, Inc), 4th edition.

Toth P, Vigo D, 2002 *An overview of vehicle routing problems. The vehicle routing problem* 1–26.

Uit het Broek MAJ, Schrotenboer AH, Jargalsaikhan B, Roodbergen KJ, Coelho LC, 2020 *Asymmetric multi-depot vehicle routing problems: Valid inequalities and a branch-and-cut algorithm. Operations Research* Forthcoming.

Van den Berg JP, 2002 *Analytic expressions for the optimal dwell point in an automated storage/retrieval system. International Journal of Production Economics* 76(1):13–25.

Van Den Berg JP, Gademann AJRM, 1999 *Optimal routing in an automated storage/retrieval system with dedicated storage. IIE transactions* 31(5):407–415.

Venkatesh P, Singh A, 2015 *Two metaheuristic approaches for the multiple traveling salesperson problem. Applied Soft Computing* 26:74–89.

Vis IFA, Roodbergen KJ, 2009 *Scheduling of container storage and retrieval. Operations Research* 57(2):456–467.

Wacholder E, Han J, Mann RC, 1989 *A neural network algorithm for the multiple traveling salesmen problem. Biological Cybernetics* 61(1):11–19.

Wauters T, Villa F, Christiaens J, Alvarez-Valdes R, Vanden Berghe G, 2016 *A decomposition approach to dual shuttle automated storage and retrieval systems. Computers & Industrial Engineering* 101:325–337.

Weidinger F, Boysen N, 2018 *Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. Transportation Science* 52(6):1412–1427.

Weidinger F, Boysen N, Briskorn D, 2018 *Storage assignment with rack-moving mobile robots in kiva warehouses. Transportation Science* 52(6):1479–1495.

Xu PJ, Allgor R, Graves SC, 2009 *Benefits of reevaluating real-time order fulfilment decisions. Manufacturing & Service Operations Management* 11(2):340–355.

Yalcin A, Koberstein A, Schocke KO, 2019 *An optimal and a heuristic algorithm for the single-item retrieval problem in puzzle-based storage systems with multiple escorts. International Journal of Production Research* 57(1):1431–65.

Yang P, Miao L, Xue Z, Ye B, 2015 *Variable neighborhood search heuristic for storage location assignment and storage/retrieval scheduling under shared storage in multi-shuttle automated storage/retrieval systems. Transportation Research Part E: Logistics and Transportation Review* 79:164–177.

Yang P, Peng Y, Ye B, Miao L, 2017 *Integrated optimization of location assignment and sequencing in multi-shuttle automated storage and retrieval systems under modified 2 n-command cycle pattern. Engineering Optimization* 49(9):1604–1620.

Yousefikhoshbakht M, Didehvar F, Rahmati F, 2013 *Modification of the ant colony optimization for solving the multiple traveling salesman problem. Romanian Journal of Information Science and Technology* 16(1):65–80.

Yu Q, Wang D, Lin D, Li Y, Wu C, 1997 *A novel two-level hybrid algorithm for multiple traveling salesman problems. Advances in Swarm Intelligence*, volume 7331 of *Lecture Notes in Computer Science*, 497–503 (Berlin Heidelberg: Springer).

Yu Y, De Koster R, 2012 *Sequencing heuristics for storing and retrieving unit loads in 3d compact automated warehousing systems. IIE Transactions* 44(2):69–87.

Yuan S, Skinner B, Huang S, Liu D, 2013 *A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. European Journal of Operational Research* 228(1):72–82.

Yuan Y, Tang L, 2017 *Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse. European Journal of Operational Research* 262(2):424–437.