High Multiplicity Asymmetric Traveling Salesman Problem With Feedback Vertex Set And Its Application To Storage/Retrieval System

Amir Gharehgozli[1], Chao Xu[2,3], Wenda Zhang[4]
[1] David Nazarian College of Business and Economics, California State University, Northridge, CA 91330
[2] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61820
[3] The Voleon Group, Berkeley, CA 94704
[4] Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61820
E-mail: amir.gharehgozli@csun.edu, the.chao.xu@gmail.com, wzhang95@illinois.edu

Abstract

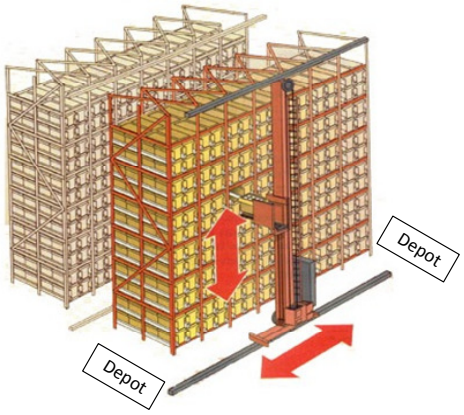
We describe an algorithm for the high multiplicity asymmetric traveling salesman problem with feedback vertex set of size k (HMATSP-kFVS) where each vertex can be visited a certain number of times and each cycle in a solution contains at least one vertex from the feedback vertex set. We show how it can be used to improve algorithms in automated storage and retrieval systems. An automated storage and retrieval system includes storage blocks and storage and retrieval machines that either move to retrieve unit loads from their current locations in the system to a depot or take unit loads from a depot and store them to specific locations in the system. Given n storage and retrieval requests in a system with k depots and one storage and retrieval machine, we show that our algorithm for HMATSP-kFVS can solve the problem of minimizing total traveling time of the storage and retrieval machine in $O(n^k + n^3)$ time when all depots are specialized (each depot fulfills one type of requests) and in $O(n^{2k} + n^3)$ time when depots are regular (each depot fulfills both types of requests). The best previous algorithm only solves the special case of the problem with 2 regular depots in $O(n^6)$ time. The applicability of our algorithm for several generalizations and special cases of the problem is also discussed. Furthermore, to evaluate the performance of our solution method, we perform extensive numerical experiments.

Keywords: Logistics; OR in Maritime Industry and Warehousing; Storage/Retrieval System; High Multiplicity ATSP; Polynomial Time Algorithm

1 Introduction

Automated storage and retrieval systems (AS/RSs) are widely used to store and retrieve products in all types of warehouses, manufacturing environments, and container terminals (Azadeh et al. 2019). There are numerous designs of AS/RS depending on the type of operations (Tappia et al. 2019, Zaerpour et al. 2013, 2019, 2015). The most common design is an S/R machine storing and retrieving unit-loads (i.e., standard containers like totes, pallets, and cartons) in between two racks. Figure 1.1 shows the top and side views of such an AS/RS. Similar designs can also be found in manufacturing environments or construction sites where an overhead crane (the S/R machine) is used to stack and retrieve raw materials, (semi-)finished products and waste instead of unit-loads. In container terminals, an AS/RS is implemented by stacking containers in a container block (stack) with an automated stacking crane (ASC) to stack and retrieve containers (Carlo et al. 2014, Gharehgozli et al. 2016, 2014).

(a) An AS/RS with storage racks and aisle-based S/R machines

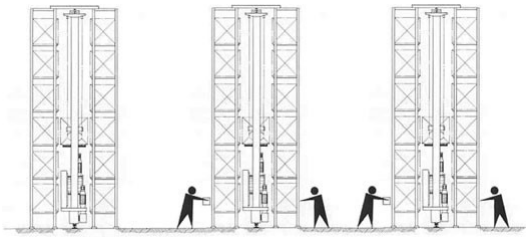


(b) Side view of a rack with two depots and an S/R machine

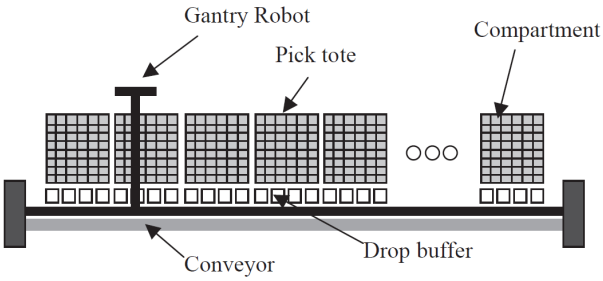
Figure 1.1. An automated storage and retrieval systems (AS/RS) with 1 depot in front and 1 at the back

In most settings, the AS/RSs have up to two depots. Therefore, the AS/RS literature has been focused mainly on two depot systems (see, for example, Gharehgozli et al. 2017b, Man et al. 2019, Yu and Yu 2019). However, settings with k depots can be also found in practice. Figure 1.2(a) shows an example of an AS/RS with Multiple In-the-Aisle Pick Positions (MIAPP-AS/RS) studied by Ramtin and Pazour (2014, 2015). The MIAPP-AS/RS is also known as a crane-supplied pick face (CSPF) system, which is analyzed by (Schwerdfeger and Boysen 2017). Figure 1.2(b) shows another example of a k depot system that has been implemented in a warehouse with 16 pick zones to receive cosmetic products from several plants in the US and overseas (Kim et al. 2003). Each pick zone has a gantry picking robot that retrieves products and drops them in 85 depots (drop buffers). Last but not the least, in container terminals, stacks can also have up to k depots, either at the ends or on the side(s) (see, for example, Galle et al. 2018, Gharehgozli et al. 2014, Li et al. 2012, 2009). An end loading stack with multiple depots is shown in Figure 1.2(c), and a side loading stack with multiple depots is shown in Figure 1.2(d).

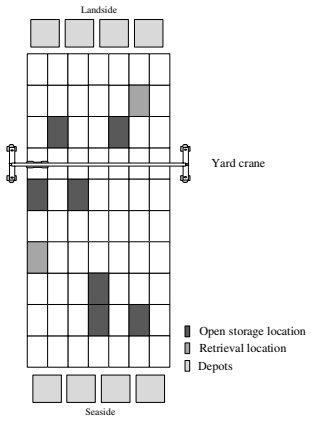
In such AS/RSs (i.e., MIAPP-AS/RS, CSPF, container stack), storage and retrieval requests are handled by the S/R machine using single and double cycles. In a single cycle, the S/R machine retrieves a unit-load from its current location to a depot or stores a unit-load from a depot to a location. In a double cycle, the S/R machine stores a unit-load to a location in the system and then moves to another unit-load to retrieve it. The problem is to sequence the storage and retrieval requests to minimize the time required to complete them all. The requests are available at the beginning and accessible (i.e., no reshuffling is required). Finally, the requests can be sequenced in any order with no precedence constraint.



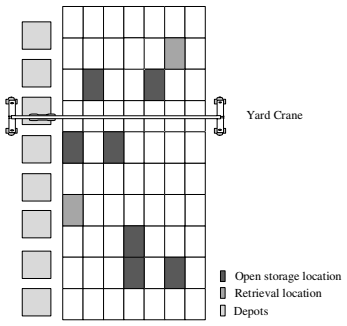
(a) A side view of an MIAPP-AS/RS



(b) A top view of a gantry picking robot with 85 depots (drop buffers) on the side



(c) A top view of an end loading stack



(d) A top view of a side loading block

Figure 1.2. AS/RSs with k depots

In general, the operational problem of sequencing storage and retrieval requests can be modeled as an asymmetric traveling salesman problem (ATSP) (Boysen and Stephan 2016). Simply stated, ATSP is a combinatorial problem in which from a given list of locations with given asymmetric pairwise travel times, a tour must be determined passing every location exactly once in such a way that the total travel time is minimized (Lawler et al. 1985). In general, ATSP is proven to be strongly $\mathcal{NP}$-hard (see, for example, Karp 1972). Although some solvable cases can be found in Burkard et al. (1998), but in general, the problem remains to be strongly $\mathcal{NP}$-hard even if special geometries, such as the Manhattan metric are applied (Boysen and Stephan 2016).

In this work, we will show that this AS/RS-related sequencing problem can be formulated as a high multiplicity asymmetric asymmetric traveling salesman problem (HMATSP) (Cosmadakis and Papadimitriou 1984, Grigoriev and Van De Klundert 2006, Sarin et al. 2011), a variant of ATSP. The problem consists of a list of locations with asymmetric pairwise travel times and a range on each location (vertex). The objective is to find a tour such that each location is traversed the number of times specified in the range and the total travel time is minimized. Note in our formulation of HMATSP, the graph does not have to be complete. Indeed, it is equivalent to the version where the graph is complete, but with infinite travel times between nodes. We consider the special case where there is a small set of nodes where each cycle in the graph must contain at least one. This set of nodes will be referred to as a feedback vertex set (FVS), and this problem will be referred to as HMATSP-kFVS when the size of the feedback vertex set is k.

Previous papers on the sequencing problem for storing and retrieving unit-loads have mainly been focused on a single or two depots, as indicated by recent reviews (Azadeh et al. 2019, Boysen and Stephan 2016, Boysen et al. 2019, Gharehgozli et al. 2016, 2019b, Gorman et al. 2014, Roodbergen and Vis 2009). Lee and Schaefer (1997) show that the optimal solution of a sequencing problem with one depot, formulated as an ATSP, can be obtained in polynomial time by relaxing subtour elimination constraints and solving the associated assignment problem (AP). Since the S/R machine returns to the same depot for every request, even if an optimal AP solution consists of multiple subtours, the S/R machine can select a subtour, carry out all requests, return to the depot, and select the next subtour.

Van den Berg and Gademann (1999) study a two depot AS/RS where one depot is dedicated to storage requests and the other one to retrieval requests. They model the problem as a transportation problem and prove that the optimal solution of the sequencing problem can be found by solving the transportation problem. Gharehgozli et al. (2017b) develop polynomial-time methods for problems with two generic depots. The authors prove that when all retrieval requests have no dedicated output depots and the S/R machine starts and ends at any of the two depots, an optimal sequence to minimize total travel time can be obtained in $O(n^6)$ steps.

There are a handful of papers working on general scenarios with more than two depots (i.e., $k > 2$ depots). One such implementation of an AS/RS can be found in warehouses (Kim et al. 2003). Most papers working on such a setting solve the problem using heuristics (Gharehgozli et al. 2015, 2017a, 2019a, Gharehgozli and Zaerpour 2018, Kim et al. 2003). On the other hand, the exact methods developed in the literature do not prove whether the problem can be solved in polynomial time. For example, Gharehgozli et al. (2014) propose a branch and bound algorithm to solve the problem. Vis and Roodbergen (2009) break down the problem into several sub-problems, each of which with one or two depots. Dynamic programming is then used to connect the solutions of these subproblems to find the optimal solution of the whole problem.

In this paper, we contribute to the AS/RS literature by developing a polynomial-time algorithm that is capable of solving the general version of the problem with k depots for any constant k, which we will refer to as SRSCHEDULE(k), and surpasses the previous algorithms in the literature. We observe that SRSCHEDULE(k) can be solved as HMATSP-kFVS. Then the algorithm can solve SRSCHEDULE(k) in $O(n^k + n^3)$ time when all depots are specialized (each depot fulfills one type of requests), and $O(n^{2k} + n^3)$ when depots are regular (each depot fulfills both types of requests).

Table 1 summarizes the performance of our algorithm on variations of SRSCHEDULE(k) compared with the best known previous results presented in Gharehgozli et al. (2017b) and Gharehgozli et al. (2014).

Table 1. Comparison with the results in Gharehgozli et al. (2017b) and Gharehgozli et al. (2014)

Problem	Previous Best	Our Result
2 specialized depot	$O(n^5)$	$O(n^3)$ Theorem 3.8
2 regular depots	$O(n^6)$	$O(n^4)$ Theorem 4.1
2 regular depots with different start and end depots	$O(n^3)$	$O(n^3)$ Theorem 4.2
2 regular depots with arbitrary start and end locations	$O(n^6)$	$O(n^5)$ Theorem 4.3
k regular depots	branch & bound	$O(n^{2k} + n^3)$ Theorem 4.1

Moreover, we contribute to the literature on solvable cases of HMATSP. Psaraftis (1980) studies HMATSP through the scheduling terminology and shows that there is a dynamic programming based algorithm where the running time is proportional to the product of the number of required visits of each node. Cosmadakis and Papadimitriou (1984) improve the previous result by proposing an algorithm with

exponential running time in the number of nodes. Grigoriev and Van De Klundert (2006) consider a new setting where all city visit requirements are increased by the same factor and describe an exponential size integer programming formulation of HMATSP. Sarin et al. (2011) improve the formulation size to polynomial. Recently, Berger et al. (2019) have developed the fastest algorithm for HMATSP in terms of both time and space complexity, which is fixed-parameter tractable when parametrized by number of nodes. The brief review of the HMATSP literature reveals that all known solution methods have exponential running times proportional to the number of nodes. We show that for graphs with a constant size feedback vertex set and bounded length between vertices in the feedback set, HMATSP is solvable in polynomial time by exploiting the feedback vertex set and min-cost circulation in the algorithm design. Our algorithm is similar to previous work, as in finding a directed tree, and then find a minimum cost tour that contains that directed tree.

The paper is organized as follows. In Section 2, we formally define HMATSP-kFVS and present the polynomial algorithm to solve it. In Section 3, we formally define SRSCHEDULE(k) and apply the algorithm developed in Section 2 to solve it. In Section 4, we present generalizations and special cases of the problem and show whether our algorithm can be adapted to solve the new problems. In Section 5, we perform extensive numerical experiments to evaluate the performance of our solution method and generate managerial insights. Finally, conclusion is given in Section 6.

2 High multiplicity traveling salesman and feedback vertex set

In this section, we first provide descriptions of graph theory concepts that will be useful in the definition of the problem and the construction of the algorithm. We then define HMATSP-kFVS and problems closely related to it. We show that a reduction exists from HMATSP-kFVS to a problem we call the min-cost F-covering $[a,b]$-circulation problem which can be solved efficiently. An algorithm that solves HMATSP-kFVS is then presented, and its improvement is discussed.

2.1 Preliminaries

A *walk* is a sequence of edges $e_1, e_2, \ldots, e_n$ such that the head of e_i is the tail of $e_{i+1}, i = 1, 2, \ldots, n-1$. A walk is an *$st$-walk* if the tail of the first edge is s and the head of the last edge is t. A walk is a *closed walk* if it is a vv-walk for some vertex v. A *path* is a walk with no repeated vertices except possibly the first and the last vertices. An *st-path* is a path starts at vertex s and ends at vertex t. A *cycle* is a path with the same start and end vertices. Given vectors $a, b \in \mathbb{N}^{|V|}$, a closed walk that visits each vertex $v \in V$ at least a_v times and at most b_v times is called an *$[a,b]$-tour*.

An undirected graph is called *connected* if there exists an st-path for every pair of vertices s and t. In a directed graph, a pair of vertices s and t is *strongly connected* if there exists a directed path from vertices s to t and a directed path from vertices t to s. A set of vertices V is *strongly connected* if it is pairwise strongly connected. A graph is called *strongly connected* if the set of all vertices is strongly connected. A directed graph is called *weakly connected* if replacing all its directed edges with undirected ones yields a connected graph. In this paper, we work with directed graphs unless it is specified otherwise.

A directed graph is a *circulation* if the in-degree (number of incoming edges) and out-degree (number of outgoing edges) are equal for each vertex in the graph. A directed graph is *Eulerian* if there exists a closed walk that uses every edge exactly once. The following theorem relates Eulerian graphs to circulations (Bang-Jensen and Gutin 2008).

Theorem 2.1 *A graph with no isolated vertices is Eulerian if and only if it is a weakly connected circulation.*

The min-cost circulation problem is the key to our representation of HMATSP-kFVS. Here we provide an overview of the problem. Let $G = (V, E)$ be a directed graph with a cost $c(e_{u,v})$ for each use of the edge $e_{u,v}$, a capacity lower bound $\ell(e_{u,v})$ and a capacity upper bound $u(e_{u,v})$ associated with each edge $e_{u,v} \in E$. The min-cost circulation problem is to find a circulation that uses edge $e_{u,v}$ at least $\ell(e_{u,v})$ times

and at most $u(e_{u,v})$ times, and then minimize the total cost. For a graph with n vertices and m edges, finding the min-cost circulation takes $O(n(m + n \log n))$ time with the successive shortest path algorithm (Ahuja et al. 1993, Chap. 9.7).

For an undirected, connected and weighted graph H, a *spanning tree* is a set with minimum number of edges that connects all vertices. A *minimum spanning tree (mst)* is a spanning tree with minimum possible total edge weight. Let mst(H) be the *weight of the minimum spanning tree* of H. If H is a directed graph, we replace all directed edges in H with undirected ones to obtain a graph H', and define mst(H) as the weight of the minimum spanning tree of H'. A set of directed edges is a *tree* if replacing all directed edges with undirected ones results in a tree.

2.2 Minimum cost tour and feedback vertex set

On a graph $G = (V, E)$ with a cost function $c : E \to \mathcal{R}$, the high multiplicity traveling salesman problem (Grigoriev and Van De Klundert 2006) can be defined as finding a minimum cost $[a, b]$-tour of G where $a, b \in \mathbb{N}^{|V|}$. The problem becomes the classic traveling salesman problem when $a = \mathbf{1}$ and $b = \mathbf{1}$.

Recall that a set of vertices A is a *feedback vertex set* (FVS) if every cycle in the graph contains at least one vertex in A. We can then define the problem HMATSP with FVS (HMATSP-kFVS) as following:

Problem 1 (HMATSP-kFVS) *Given a directed graph $G = (V, E)$, a size k FVS $A \subseteq V$, a cost function $c : E \to \mathcal{R}$ and vectors $a, b \in \mathbb{N}^{|V|}$, find a minimum cost $[a, b]$-tour.*

If we require all vertices in the FVS to be visited at least once, we have the following variation of HMATSP-kFVS:

Problem 2 (Variation of HMATSP-kFVS) *Given a directed graph $G = (V, E)$, a size k FVS $A \subseteq V$, a cost function $c : E \to \mathcal{R}$ and vectors $a, b \in \mathbb{N}^{|V|}$ such that $a_v \geq 1$ for all vertices $v \in A$, find a minimum cost $[a, b]$-tour.*

We then have the following relation between the two problems.

Theorem 2.2 HMATSP-kFVS *reduces to $O(2^k)$ instances of Problem 2.*

Proof: Given an instance of HMATSP-kFVS, let $A_0 = \{v | v \in A, a_v = 0\}$. For each $S \subseteq A_0$, we solve Problem 2 for induced subgraph $G[(V \setminus A_0) \cup S]$ with feedback vertex set S. The minimum of all solutions is the desired solution for HMATSP-kFVS. □

In this work we assume all FVSs to have constant sizes. As a result, HMATSP-kFVS and Problem 2 are polynomial time equivalent based on Theorem 2.2.

2.3 An algorithm to solve HMATSP-kFVS

To solve HMATSP-kFVS and Problem 2, we first need to discuss the following problems.

Problem 3 (min-cost F-covering $[a, b]$-Circulation Problem) *Given a directed graph $G = (V, E)$, a cost function $c : E \to \mathcal{R}$, vectors $a, b \in \mathbb{N}^{|V|}$ and a subset of edges $F \subseteq E$, find a minimum cost F-covering $[a, b]$-circulation.*

We first build an auxiliary graph G' from G by applying the standard split vertex operation on each vertex in V. For each vertex $v \in V$, we create v^-, v^+ and an edge e_{v^-,v^+}. Then for each edge $e_{u,v}$, we create an edge e_{u^+,v^-}. The cost of each edge e_{u^+,v^-} is $c(e_{u,v})$. The cost of each edge e_{v^-,v^+} is 0. Let each edge e_{v^-,v^+} have capacity lower bound a_v and upper bound b_v; let each edge in F (if $e_{u,v}$ in F, the corresponding edge is e_{u^+,v^-}) have capacity lower bound 1.

It can be observed that the min-cost circulation for the auxiliary graph G' is a solution to Problem 3. Next, we show that there exists a reduction from Problem 2 to Problem 3.

A set of edges F is an *A-connector*, if induced subgraph $G[F]$ is a connected graph that contains all vertices in A. A *min-cost F-covering $[a,b]$-tour* is an F-covering $[a,b]$-tour of minimum cost. If F is an A-connector, then the min-cost F-covering $[a,b]$-tour can be found quickly.

Theorem 2.3 *Let A be a feedback vertex set of G, and $a_v \geq 1$ for all $v \in A$. Let F be an A-connector. A min-cost F-covering $[a,b]$-circulation of G is a min-cost F-covering $[a,b]$-tour.*

Proof: Let W be a min-cost F-covering $[a,b]$-tour. Let C be the min-cost F-covering $[a,b]$-circulation of G. Every $[a,b]$-tour that contains F is an F-covering $[a,b]$-circulation, therefore cost of C is at most cost of W. We will show that C is connected, which implies that it is an $[a,b]$-tour, and shows cost of C equals cost of W. Every cycle contains a vertex in A, and every vertex in the circulation is in a cycle. Then every vertex in C is connected to some vertex in A. Moreover, F connects all vertices in A and $F \subseteq C$. This shows that C is connected. □

By the above theorem, we quickly obtain a simple algorithm: find min-cost F-covering $[a,b]$-tour for each A-connected F, the minimum one is the min-cost $[a,b]$-tour. Of course, one can refine it to only consider a much smaller set of A-connectors. To capture this family of A-connectors, we will introduce generating families.

Given a directed graph $G=(V,E)$ and an FVS $A \subseteq V$, let $\mathcal{W}$ be the family of $[a,b]$-tours and let $\mathcal{F}$ be a family of A-connectors. Then $\mathcal{F}$ is called a generating family if there exists some $F \in \mathcal{F}$ such that $F \subseteq W$ for each $W \in \mathcal{W}$.

Theorem 2.4 *On a directed graph $G=(V,E)$ with an FVS $A \subseteq V$ from Problem 2, if $\mathcal{F}$ is a generating family, there exists an algorithm that solves Problem 2 by solving the min-cost F-covering $[a,b]$-circulation problem for each $F \in \mathcal{F}$.*

Proof: Consider an algorithm that solves for the min-cost F-covering $[a,b]$-circulation for each $F \in \mathcal{F}$ and returns the minimum. Let W^* be an optimal solution to Problem 2. There is an $F \in \mathcal{F}$ such that $F \subseteq W^*$. By Theorem 2.3, the returned solution has cost no more than the cost of W^*. Therefore the algorithm returns an optimal solution to Problem 2. □

We get the following corollary directly from the above theorem.

Corollary 2.5 *On a directed graph $G=(V,E)$ with an FVS $A \subseteq V$ from Problem 2, if $\mathcal{F}$ is a generating family, there exists an algorithm that solves Problem 2 in $O(|\mathcal{F}|n^3)$ time.*

Corollary 2.5 can be proved by following the steps of the algorithm shown in Figure 2.1. Since HMATSP-kFVS and Problem 2 are polynomial time equivalent, the algorithm also solves HMATSP-kFVS in $O(|\mathcal{F}|n^3)$ time, because it calls the min-cost circulation algorithm $|\mathcal{F}|$ times.

```
BuildAuxiliaryGraph(G = (V, E), a, b, c):
    V' ← {v⁻, v⁺ | v ∈ V}
    E' ← {e_{v⁻,v⁺} | v ∈ V} ∪ {e_{u⁺,v⁻} | e_{u,v} ∈ E}
    for e_{u,v} ∈ E:
        c'(e_{u⁺,v⁻}) ← c(e_{u,v})
    for v ∈ V:
        c'(e_{v⁻,v⁺}) ← 0
        ℓ(e_{v⁻,v⁺}) ← a_v
        u(e_{v⁻,v⁺}) ← b_v
    return G' = (V', E'), cost c', capacity lower bound ℓ, upper bound u.

HMTSPFVSFromGeneratingFamily(G, F, a, b, c):
    G', c', ℓ, u ← BuildAuxiliaryGraph(G, a, b, c)
    for each F ∈ F
        ℓ' ← ℓ
        for each e_{u,v} ∈ F
            ℓ'(e_{u⁺,v⁻}) ← 1
        C_F ← MinCostCirculation(G', c', ℓ', u)
    return C_F of minimum cost
```

Figure 2.1. Solving the HMATSP-kFVS, given that $\mathcal{F}$ is a generating family of G. MinCostCirculation(G, c, ℓ, u) takes a graph $G = (V, E)$, a cost function $c : E \to \mathcal{R}$, and lower and upper capacity bound functions $\ell, u : E \to \mathbb{N}$, and returns a min-cost circulation.

2.4 Improve the algorithm

The algorithm discussed in the previous section requires solving the min-cost circulation problem repeatedly and the running time is $|\mathcal{F}|$ calls of min-cost circulation algorithm. However, successive calls to the min-cost circulation problem are related. We can exploit this fact to get a faster running time by considering the following problem:

Problem 4 (Dynamic min-cost circulation problem) *Given a solution to the min-cost circulation problem, with an update to the lower bound of a single edge by* 1*, find the min-cost circulation on the modified graph.*

The new min-cost circulation for this problem can be processed by a single shortest path computation on the residual graph for the original min-cost circulation (Ahuja et al. 1993, Chap. 9.11). The following theorem gives the computational complexity of that operation.

Theorem 2.6 *For a graph with n vertices and m edges, after one min-cost circulation computation, each dynamic min-cost circulation problem can be solved in $O(m + n \log n)$ time.*

For any sets of edges F, the following theorem shows the computational complexity of finding the solution to Problem 3 for F if we already know the solution to Problem 3 for $F = \emptyset$.

Theorem 2.7 *Let F be a set of edges. Given an optimal solution to Problem 3 for $F = \emptyset$, we can find the optimal solution to Problem 3 for F in $O(|F|(m + n \log n))$ time.*

Proof: We increase the lower bound for each edge in F to 1. Each operation can be done by solving Problem 4. By Theorem 2.6, we have the overall computational complexity. □

We now have an improved algorithm for Problem 2 and HMATSP-kFVS. The algorithm checks each $F \in \mathcal{F}$ as described in Theorem 2.7 and returns the one with the minimum cost. Its computational complexity is as indicated in the following theorem.

Theorem 2.8 *Let $\mathcal{F}$ be a generating family. There exists an algorithm that solves Problem 2 in $O(n^2 \sum_{F\in\mathcal{F}} |F| + n^3)$ time.*

Proof: See the algorithm in Figure 2.2. The initial solution to the min-cost circulation problem takes n^3 time and then apply Theorem 2.7 repeatedly to every set in $\mathcal{F}$ to get the result. □

We remark the algorithm is only an improvement when $n|\mathcal{F}| \geq \sum_{F\in\mathcal{F}} |F|$. Namely, only when A-connectors in $\mathcal{F}$ are small. This is indeed the case, as for our applications, all A-connectors consists of a constant number of edges.

```
HMTSPFVSFromGeneratingFamily(G, F, a, b, c):
  G', c', ℓ, u ← BuildAuxiliaryGraph(G, a, b, c)
  C ← MinCostCirculation(G', c', ℓ, u)
  for each F ∈ F:
      C_F ← C
      ℓ' ← ℓ
      for each e_{u,v} ∈ F:
          C_F ← DynamicMinCostCirculation(C_F, e_{u⁻,v⁺}, G', c', ℓ', u)
          ℓ'(e_{u⁻,v⁺}) ← ℓ'(e_{u⁻,v⁺}) + 1
  return C_F of minimum cost
```

Figure 2.2. An improved algorithm compared to Figure 2.1. DynamicMinCostCirculation takes an instance of min-cost circulation, an optimal circulation, and an edge. It outputs the new min-cost circulation given we increase the lower bound of the edge by 1.

2.5 Size of a generating family

To find out the computational complexity for the algorithm that solves Problem 2, the size of the generating family $\mathcal{F}$ remains to be shown. In this section, we bound the size of a generating family by bounding the weight of spanning trees in a graph on the FVS which we call the condensed graph. First, we present some definitions that will help the discussions in this section.

Let A be an FVS of G. A path is *internally A-avoiding* if no internal vertices are in A. For each $u, v \in A$, we define $\rho(u, v)$ to be the maximum number of internal vertices in an internally A-avoiding uv-path in G. The graph $H = (A, E)$ is the *condensed graph* of G and an edge $e_{u,v}, u, v \in A$ is in H if and only if there exists an internally A-avoiding uv-path in G. The weight of an edge $w(e_{u,v})$ in the condensed graph H is $\rho(u, v)$. For a set of edges E, we abuse the notation and write $w(E)$ to denoted $\sum_{e\in E} w(e)$. A set of edges F is *topologically equivalent* to a tree T on A if it is a union of some internally A-avoiding uv-path for each edge $e_{u,v} \in T$.

Given a graph $H = (A, E)$, we define a set of spanning trees $\mathcal{T}$ on H as *complete* if every strongly-connected subgraph of H contains at least one tree in $\mathcal{T}$ as a spanning tree. We first show an upper bound on the size of a complete family of spanning trees.

Lemma 2.9 (Size of a complete family of spanning trees) *A complete family of spanning trees on a set of k vertices has at most $2^{k-1}k^{k-2}$ trees.*

Proof: The number of distinct undirected trees on k vertices is k^{k-2} by Cayley's formula (Cayley 1889). Each tree has $k-1$ edges, and there can be 2 directions for each edge. Hence there can be at most $2^{k-1}k^{k-2}$ directed trees on k vertices. The number of trees is clearly an upper bound for a complete family of spanning trees. □

The following theorem connects the size of a complete family of the condensed graph of G and the size of a generating family of G. We call $\max_{T\in\mathcal{T}} w(T)$ to be the *toughness* of $\mathcal{T}$.

Theorem 2.10 *Let k be a constant. Given a graph G with an FVS A of size k. H is the condensed graph of G with respect to A. Let T be a spanning tree of H with weight t. The set of edges topologically equivalent to T is $O(n^t)$.*

Proof: Let $\tau(u,v)$ to be the number of internally A-avoiding uv-paths in G. The number of sets of edges topologically equivalent to T is at bounded by

$$\begin{aligned}\prod_{e_{u,v}\in T}\tau(u,v) &\leq \prod_{e_{u,v}\in T} n^{w(e_{u,v})}\\ &= n^{\sum_{e_{u,v}\in T} w(e_{u,v})}\\ &= n^t\end{aligned}$$

□

Theorem 2.11 *Let k be a constant. Given a graph G with an FVS A of size k. Let $\mathcal{T}$ be a complete family of spanning trees on the condensed graph H of G with toughness t. There exists a generating family $\mathcal{F}$ on G with size $O(n^t)$.*

Proof: Let $\mathcal{F}_T$ be set of set of edges that are topologically equivalent to $T \in \mathcal{T}$. Let $\mathcal{F} = \bigcup_{T\in\mathcal{T}} \mathcal{F}_T$, we will show this is the desired generating family.

By Theorem 2.10. $|\mathcal{F}_T| \leq n^t$ topologically equivalent sets. $|\mathcal{F}| \leq |\mathcal{T}| n^t \leq 2^{k-1}k^{k-2}n^t = O(n^t)$.

We have to show that $\mathcal{F}$ is a generating family. Let W be a walk in G. Note that W induces a strongly-connected graph in H by replacing each internally A-avoiding uv-path in W where $u, v \in A$ with the edge $e_{u,v}$. This graph is denoted as H_W. By definition of $\mathcal{T}$, there exists a $T \in \mathcal{T}$ that is a spanning tree of H_W. Construct a set $F \in \mathcal{F}_T$ as follows: for each edge $e_{u,v}$ in T, take an arbitrary internally A-avoiding uv-path in W. One can see $F \subseteq W$. Because every $F \in \mathcal{F}$ connects A, therefore, $\mathcal{F}$ is a generating family. □

Let $\mathcal{H}$ be the set of strongly-connected subgraphs of H. We define the ideal complete family as

$$\mathcal{T} = \left\{T \text{ is a minimum spanning tree in } H' \mid H' \in \mathcal{H}\right\}.$$

The ideal complete family is complete family with smallest toughness. For graph of constant size, the ideal complete family can be found in constant time. We are now ready to return to our improved algorithm for solving HMATSP-kFVS.

Theorem 2.12 *Let k be a constant. Given a graph G with an FVS A of size k, let H to be the condensed graph of G. If the ideal complete family of H has toughness t, then there exists a generating family of $O(n^t)$ trees, and there exists an algorithm that solves HMATSP-kFVS in $O(n^{t+2} + n^3)$ time.*

Proof: By Theorem 2.11, we know there is a generating family with size at most $O(n^t)$. The computational complexity follows from Theorem 2.8. □

For any input graph of HMATSP-kFVS, we can find an upper bound over all minimum spanning trees of each strongly-connected graph on A and apply the previous theorem.

```
TopologicallyEquivalent(G = (V, E), T):
    Π ← all permutations of w(T) elements of V
    e_{u_1,v_1}, ..., e_{u_k,v_k} are the edges of T
    for X ∈ Π:
        for i ∈ {1, ..., k}:
            j ← w(e_{u_i,v_i})
            z_1, ..., z_j ← the next j elements from X
            P_i ← the edges on the path u_i, z_1, ..., z_j, v_i
        F_X ← ⋃_{i=1}^{k} P_i
    F ← {F_X | X ∈ Π}
    return F

HMTSPFVS(G, A, a, b, c):
    H ← condensed graph of G with respect to A
    T ← ideal complete family of H
    F ← ⋃_{T∈T} TopologicallyEquivalent(G, T)
    return HMTSPFVSFromGeneratingFamily(G, F, a, b, c)
```

Figure 2.3. Solving the HMATSP-kFVS on G given an FVS A using HMTSPFVS(G, A, a, b, c), where a, b are vertex visit lower and upper bounds, and c is the edge cost function.

3 The Application of HMATSP-kFVS to SRSCHEDULE(k)

In this section, we show that SRSCHEDULE(k) can be reduced to HMATSP-kFVS and thus can be solved by the algorithm discussed in Section 2. First, we formally define the problem SRSCHEDULE(k). Then the graph representation of the problem is presented and the computational complexity of applying our algorithm is discussed.

3.1 Problem Definition

Let S and R be the sets of locations for the storage (input) and retrieval (output) requests, respectively; let $|S \cup R| = n$ and $S \cap R = \emptyset$. Let D_I and D_O be the sets of locations for the input and output depots, respectively; let $D = D_I \cup D_O$, $D_I \cap D_O = \emptyset$ and $|D| = k$. In this section, we assume all depots are either input-only or output-only. They are called *specialized* depots while the depots that fulfill both types of requests are called *regular* depots. The scenarios with regular depots will be discussed in Section 4.

Each input request consists of (D_{s_i}, s_i) where $D_{s_i} \subseteq D_I$ is a set of input depots for the S/R machine to pick up the item and $s_i \in S$ is the location of the request. Similarly, each output request consists of (D_{r_i}, r) where $D_{r_i} \subseteq D_O$ is a set of output depots for the S/R machine to drop off the item, and $r_i \in R$ is the location of the request.

Let $L = S \cup R \cup D$ be the set of all locations. A distance function dist : $L \times L \to \mathcal{R}^+$ that satisfies the triangle inequality $\text{dist}(x, z) \leq \text{dist}(x, y) + \text{dist}(y, z)$ for all $x, y, z \in L$ is provided. In Gharehgozli et al. (2017b), the distance is the Chebyshev distance, but more general distance function can be applied.

A single start location ℓ_s and a single end location ℓ_t are provided as well. In this section, we assume that $\ell_s, \ell_t \in D$ and refer to them as d_s and d_t. The S/R machine must start from the depot d_s, complete all S/R operations and stop at the depot location d_t. Figure 3.1 shows the notations discussed in this section.

The objective is to find a sequence for the S/R requests that minimizes the total operation time. Assume constant movement speed for the S/R machine and the objective can also be viewed as finding a sequence of S/R requests that minimizes the total distance (in time) traversed by the S/R machine.

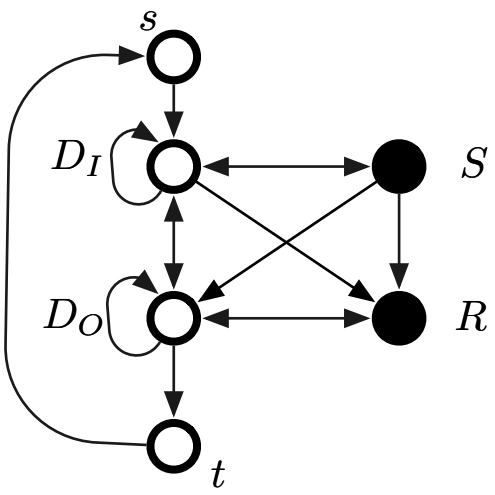


Figure 3.1. SRschedule(k) where d_s is a vertex in D_I and d_t is a vertex in D_O

3.2 Reduce to graph problem

Define a directed graph $G = (V, E)$ and let the set of vertices be $V = L \cup \{s, t\}$. The edge set E includes the following edges:

1. There is an edge e_{d,s_i} for each storage request $s_i \in S$ and $d \in D_{s_i}$.
2. There is an edge $e_{s_i,d}$ for each storage request $s_i \in S$ and $d \in D$.
3. There is an edge e_{d,r_i} for each retrieval request $r_i \in R$ and $d \in D$.
4. There is an edge $e_{r_i,d}$ for each retrieval request $r_i \in R$ and $d \in D_{r_i}$.
5. There is an edge between each pair of depots $d, d' \in D$.
6. There is an edge e_{s_i,r_j} for each storage request $s_i \in S$ and retrieval request $r_j \in R$.
7. There is an edge e_{s,d_s} and an edge $e_{d_t,t}$.
8. There is an edge $e_{t,s}$ from t to s.

We define a cost function $c : E \to \mathcal{R}$ to each edge $e \in E$. In particular, $c(e_{u,v}) = \text{dist}(u, v)$ for all vertices $u, v \in L$ and $c(e_{u,v}) = 0$ if either u or v is s or t. We are interested in finding a minimum cost walk that starts from vertex s, ends at vertex t and visits every vertices in $S \cup R$.

Let the vertex set D which represents the depots be the feedback vertex set. This problem is a special case of HMATSP-kFVS where $a_v = 1$ for all vertices $v \in S \cup R$, $a_v = b_v = 1$ for all vertices $v \in \{s, t\}$ and $a_v = 0$ for all vertices $v \in D$. The computational complexity of our algorithm in solving this problem can be found by applying Theorem 2.12. However, we still need to bound the toughness of the ideal complete family $\mathcal{T}$.

To bound the toughness of $\mathcal{T}$, we first develop the following 2-colored graph to represent the two types of depots. A graph is called 2-*colored* if the vertices have either a color 0 or 1. Let $H = (V, E)$ be a strongly connected 2-colored directed graph on k vertices. Define vertex weight functions w_0 and w_1 where $w_i(v) = 0$ if i is the color of the vertex v and $w_i(v) = 1$ otherwise. As a result, we have $w_0(v) + w_1(v) = 1$ for all vertices $v \in V$. Let each edge $e \in E$ have a weight $w(e_{u,v}) = w_0(u) + w_1(v)$. The colors indicate if a depot is output-only (0) or input-only (1). The weight of an edge $e_{u,v}$ in H is an

upper bound to the maximum number of non-depot vertices traversed moving from depots u to v in G. The graph H can be interpreted as a subgraph of the condensed graph of G. We are then interested in bounding $\text{mst}(H)$.

Theorem 3.1 *Let C be a 2-colored directed cycle on k vertices. The total weight is k.*

Proof: For a directed cycle, a vertex has an edge going in and an edge going out. Let V be the vertices in C. Each vertex with color 0 or 1 adds weight 1 for one edge. The total edge weight is then $\sum_{e\in E} w(e) = \sum_{v\in V}(w_0(v)+w_1(v)) = |V| = k$. □

Theorem 3.2 *Let P be a directed st-path on k vertices where $s \neq t$. The total weight of P is $k-w_1(s)-w_0(t)$.*

Proof: Add an edge from t to s in P, and we obtain a directed cycle. Use Theorem 3.1, we know the weight of P is $k - w(e_{t,s}) = k - w_1(s) - w_0(t)$. □

Theorem 3.3 *Given a directed 2-colored st-path on k vertices P where $s \neq t$, if $w_0(s) = 1$ and $w_1(t) = 1$, then P contains an edge of weight 2.*

Proof: Let the sequence of vertices in P be $s = v_1, v_2, \ldots, v_n = t$. Assume that there is no edge with weight 2. If $w_0(v_i) = 1$, then $w_1(v_{i+1}) = 0$. Otherwise, the edge $e_{v_i,v_{i+1}}$ has weight 2. Since $w_0(v)+w_1(v) = 1$, it also implies that $w_0(v_{i+1}) = 1$. By induction, we can derive similar result for every vertex v as $w_1(v)$ is 0 for all v. A contradiction to $w_1(v_n) = 1$. □

It is necessary to take into consideration the special case where all vertices are of the same color. Given a 2-colored graph $G = (V, E)$, define a function $\gamma(\cdot)$ such that $\gamma(G)$ is the number of distinct colors in G. We then have the following theorem.

Theorem 3.4 *Let $C = (V, E)$ be a k-vertex 2-colored directed cycle, then* $\text{mst}(C) = k - \gamma(C)$.

Proof: Since every spanning tree of C is a path that avoids one edge in C, if the maximum edge weight is w, then $\text{mst}(C) = k - w$.

If $\gamma(C) = 1$, then $C = (V, E)$ is not a k-vertex 2-colored directed cycle. Every edge in C has weight 1, and $\text{mst}(C) = k - \gamma(C)$ is true. On the other hand, if $\gamma(C) = 2$, then $C = (V, E)$ is a k-vertex 2-colored directed cycle. There has to be an edge going from a color 0 vertex to a color 1 vertex that has weight 2. As a result, $\text{mst}(C) = k - \gamma(C)$ is also true. □

Before the next theorem, it is necessary to introduce additional graph theory background. An *ear* is a directed path of which the internal vertices each has an in-degree and out-degree of 1. An *ear decomposition* of a strongly connected graph is a sequence of ears $E_1, E_2, \ldots, E_t$ that partitions the edges of E. Ear E_i only share its end vertices with $E_1, E_2, \ldots, E_{i-1}$. None of its internal vertices is in $E_1, E_2, \ldots, E_{i-1}$.

It is known that every strongly connected graph has an ear decomposition.

Theorem 3.5 (Bang-Jensen and Gutin (2008)) *For a strongly connected graph G and a directed cycle C in G, there exists an ear decomposition $E_1, E_2, \ldots, E_t$, such that $E_1 = C$.*

Lemma 3.6 *In a strongly connected 2-colored graph G, if $\gamma(G) = 2$, then there exists a cycle with at least one vertex of each color.*

Proof: Consider an ear decomposition $E_1, \ldots, E_k$ of G. Let E_i be the first ear that contains vertices of two different colors. If $i = 1$, then E_i is a cycle with at least one vertex of each color. If $i > 1$, then it is an st-path. Let P be any ts-path in $E_1 \cup \ldots \cup E_{i-1}$ and $P \cup E_i$ is a cycle with at least 2 colors. □

The weight of a minimum spanning trees on a strongly-connected graph has the following upper bound.

Theorem 3.7 *Let $H = (V, E)$ to be a strongly connected 2-colored graph, then* $\text{mst}(H) \leq |V| - \gamma(H)$.

Proof: We prove the theorem by induction on the number of ears in the ear decomposition.

Let H have an ear decomposition $H_1, H_2, \ldots, H_t$ where $\gamma(H_1) = \gamma(H)$. Such ear decomposition exists due to Theorem 3.5. If $\gamma(H) = 1$, then $\gamma(H_1) = 1$ as all vertices in H_1 must have the same color. If $\gamma(H) = 2$, then by Lemma 3.6 there exists a cycle that contains vertices of both colors. Let that cycle be H_1.

The base case with 1 ear is a cycle, the result of which is implied by Theorem 3.4.

Assume that $\text{mst}(G) \leq |V(G)| - \gamma(G)$ holds for any graph G with an ear decomposition of at most $t-1$ ears. Let H' be $\bigcup_{i=1}^{t-1} H_i$. The choice of H_1 makes sure that $\gamma(H) = \gamma(H')$. Note that H_t is a uv-path where both u and v are in H'. All other vertices in H_t does not appear in H'. Let e' be the maximum weight edge in H_t.

Depending on whether $u = v$, there are two cases.

Case 1. If $u = v$, then $\sum_{e \in E(H_t)} w(e) = |V(H_t)|$ and we have the following inequalities:

$$\begin{aligned}
\text{mst}(H) &\leq \text{mst}(H') + \text{mst}(H_t) \\
&= \text{mst}(H') + |V(H_t)| - w(e') && \text{[mst of a cycle]} \\
&\leq \big(|V(H')| - \gamma(H')\big) + |V(H_t)| - w(e') && \text{[inductive hypothesis]} \\
&= (|V(H)| - \gamma(H)) + 1 - w(e') && [|V(H)| = |V(H')| + |V(H_t)| - 1] \\
&\leq |V(H)| - \gamma(H) && [w(e') \geq 1 \text{ for cycles}].
\end{aligned}$$

Case 2. If $u \neq v$, then $\sum_{e \in E(H_t)} w(e) = |V(H_t)| - w_1(u) - w_0(v)$ and we have the following inequalities:

$$\begin{aligned}
\text{mst}(H) &\leq \text{mst}(H') + \sum_{e \in E(H_t)} w(e) - w(e') \\
&\leq \big(|V(H')| - \gamma(H')\big) + \big(|V(H_t)| - w_1(u) - w_0(v) - w(e')\big) && \text{[weight of a path]} \\
&= (|V(H)| - \gamma(H)) + 2 - w_1(u) - w_0(v) - w(e') && [|V(H)| = |V(H')| + |V(H_t)| - 2].
\end{aligned}$$

It suffices to show that $w_1(u) + w_0(v) + w(e') \geq 2$. If $w_1(u) + w_0(v) = 2$, the inequality holds. If $w_1(u) + w_0(v) = 1$, then the edge containing u or the edge containing v would have weight at least 1. Therefore $w(e') \geq 1$ and the inequality holds. Finally, if $w_1(u) + w_0(v) = 0$, then $w_0(u) = w_1(v) = 1$. Theorem 3.3 indicates that $w(e') = 2$ and the inequality holds. □

Finally, we can use our results on strongly connected 2-colored graph and apply it to the SRSCHEDULE(k) problem.

Theorem 3.8 SRSCHEDULE(k) *can be solved in $O(n^k + n^3)$ if there is at least one depot of each type. Otherwise it can be solved in $O(n^{k+1} + n^3)$.*

Proof: By invoking Theorem 2.12 and Theorem 3.7, if there is at least one depot of each type, we know that there exists an $O(n^k + n^3)$ time algorithm for SRSCHEDULE(k). Otherwise there exists an algorithm that runs in $O(n^{k+1} + n^3)$ time. □

4 Extensions to SRSCHEDULE(k)

In this section, we show that our algorithm can solve the variations to SRSCHEDULE(k) discussed in Gharehgozli et al. (2017b) and the performance matches or improves from previous works.

4.1 Depots accept both input and output

Gharehgozli et al. (2017b) assume that all depots are regular. Figure 4.1 shows an example of the problem with regular depots. To solve this variation of SRSCHEDULE(k), we construct the same graph as in section 3.2 but with $D_O = D_I$. One can see that $\rho(u,v) \leq 2$ for all $u, v \in D$. We obtain the following theorem because every spanning tree in the condensed graph has weight no more than $2(k-1)$.

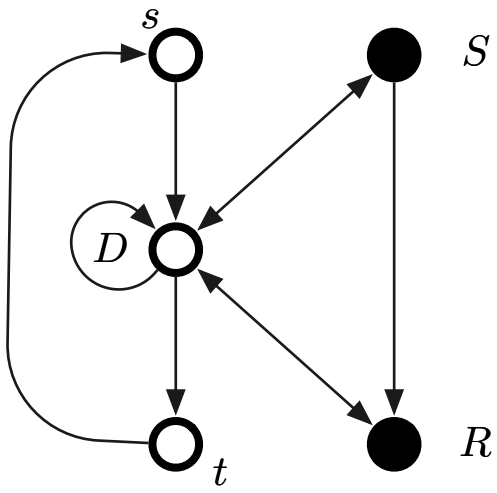


Figure 4.1. SRschedule(k) with only regular depots

Theorem 4.1 SRSCHEDULE*(k) with k regular depots can be solved in $O(n^{2k}+n^3)$.*

Proof: Each regular depot is an input depot and an output depot. It is the same as solving SRSCHEDULE($2k$) with k input depots and k output depots. By Theorem 3.8, it takes $O(n^{2k}+n^3)$ time. □

In the case of two depots, if the start and end depots are the same, our result is $O(n^4)$ which improves from the result of $O(n^6)$ by Gharehgozli et al. (2017b).

On the other hand, Gharehgozli et al. (2017b) also propose an $O(n^3)$ time algorithm in the case of two regular depots where the start and end depots are different. We now consider our algorithm specifically for this case.

Theorem 4.2 SRSCHEDULE*(k) with k regular depots can be solved in $O(n^{2k-2}+n^3)$ if the start and end depots are different.*

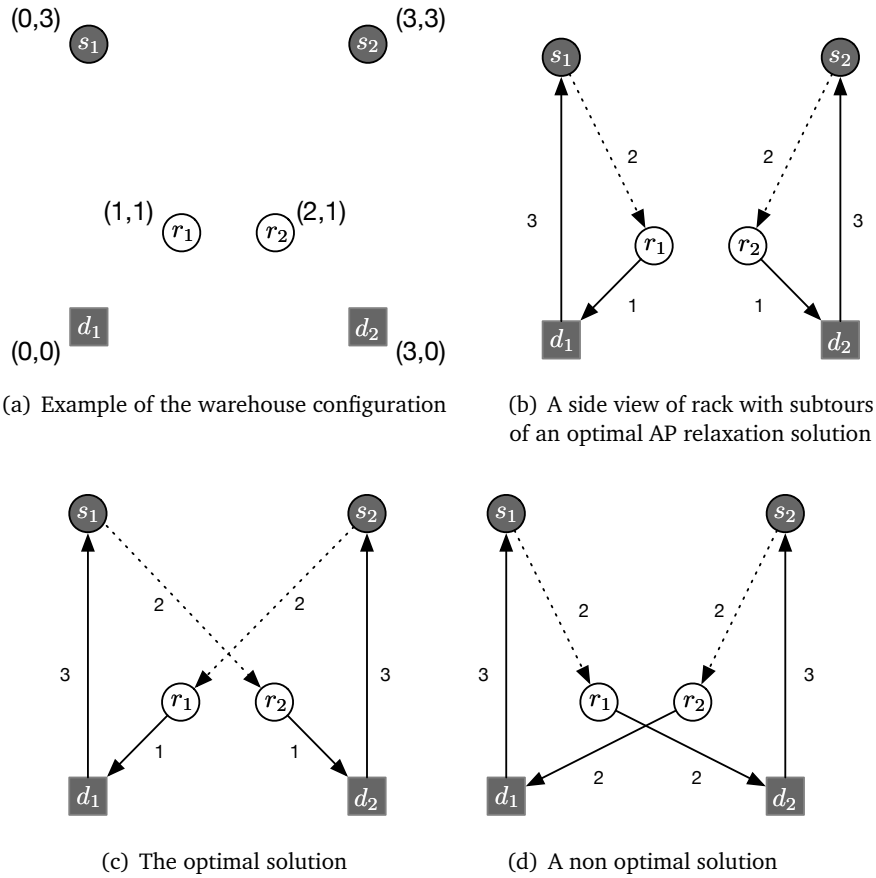
Proof: Let $A = D \cup \{s,t\}$ and let H be the condensed graph of G. For each strongly-connected subgraph H' of H, there is a spanning tree in H' with weight $2k-4$. We can construct a T in the condensed graph and it has weight $2k-2$. Note that every tree in H' must contain 3 edges $e_{s,d_s}, e_{t,s}, e_{d_t,t}$. All 3 edges has weight 0 and $d_s \neq d_t$. We can contract s, t, d_s, d_t into a single vertex u and obtain a graph H''. Because there are $k-1$ vertices in H'' and the edge weights between any two vertices is at most 2, any tree in H'' has weight at most $2k-4$. Take that tree, uncontract the vertices, add edges $e_{s,d_s}, e_{t,s}, e_{d_t,t}$, and we obtain a set of edges with weight at most $2k-4$ that connects H'. Therefore, H' has a spanning tree with weight no more than $2k-4$. We obtain the theorem by then applying Theorem 2.12. □

This matches the computational complexity of $O(n^3)$ and extends the algorithm to the case of k depots.

We use the example shown in Figure 4.2a to explain the $O(n^{2k}+n^3)$ steps proved in Theorem 4.1. The other algorithms can be followed similarly. We first describe the inputs. There are $k=2$ depots d_1 and

d_2. There are $n = 4$ requests, two of which are storage requests, a storage request d_1 to s_1 and a storage request d_2 to s_2. There are also two retrieval requests, r_1 to any depot, and r_2 to any depot. The locations are $d_1 = (0,0)$, $d_2 = (3,0)$, $s_1 = (0,3)$, $s_2 = (3,3)$, $r_1 = (1,1)$, $r_2 = (2,1)$. For ease of calculation, the speed of the crane is 1 in each direction, and takes time 0 to pick up the item. Therefore, the travel time for the crane to move from (x, y) to (x', y') and pickup and dropoff is $\max(|x - x'|, |y - y'|)$. The instance requires the start and end location to be at d_1.

Figure 4.2b shows a side view of a rack (or a top view of a container block) with an optimal AP relaxation solution with cost 12, which contains two subtours, so it does not solve the desired problem. Figure 4.2c shows that the optimal tour for the crane visits the following locations in sequence $d_1, s_1, r_2, d_2, s_2, r_1, d_1$. The total time is 12. Note that it is no worse than the optimal AP relaxation solution. A non optimal solution, where the total time is 14, is shown in Figure 4.2d. For simplicity, we are assuming that the solution has to visit all depots. This is the case for our inputs. Otherwise there is an extra step of trying all possible subset of visited depots.



(a) Example of the warehouse configuration

(b) A side view of rack with subtours of an optimal AP relaxation solution

(c) The optimal solution

(d) A non optimal solution

Figure 4.2. An schematic example to show the steps of the algorithm

The first step is to reduce the depot problem to HMATSP-kFVS. The input graph for HMATSP-kFVS is shown in Figure 4.3(a). We call it graph G. In G, we are interested in finding a minimum cost tour that visits all vertices at least once, where the FVS we consider is $A = \{d_1, d_2, s, t\}$. The cost of an edge between two depots is the time it takes to move from one to the other. The cost of edges incident to s and t is 0. The optimal solution is shown in Figure 4.3(b). Let the tour be S. By tracing S starting from s, and considering the corresponding operations in SRSCHEDULE(k), we obtain the optimal solution to the original problem. We will derive S in the rest of this section.

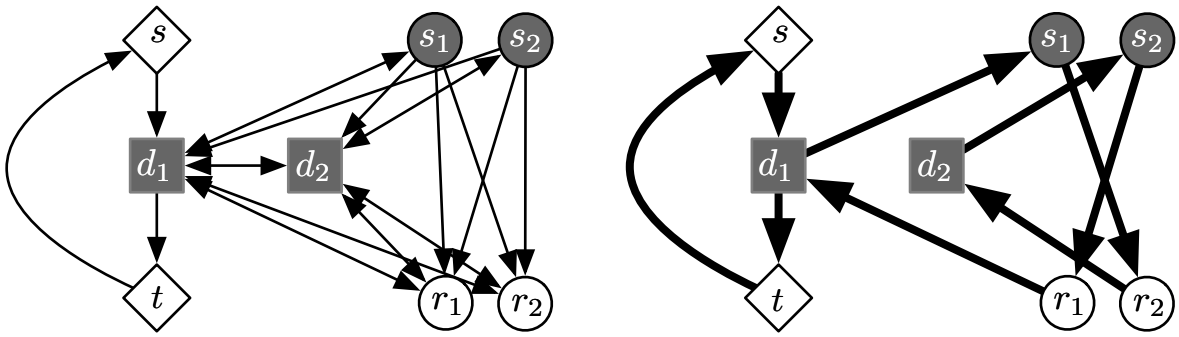


Figure 4.3. The instance for HMATSP-kFVS and an optimal solution S for the instance.

In order to solve HMATSP-kFVS, we need to consider the condensed graph. The condensed graph of G, which we call H, is simply on the vertices in A, and for $a, b \in A$, there is an edge from a to b if there is a path from a to b without visiting any other vertex in A. The condensed graph H of our example is shown in Figure 4.4(a). In the graph, the weight of edge e_{d_1,d_2} and e_{d_2,d_1} is 2, and 0 for all other edges. The condensed graph H is the unique strongly connected subgraph that contains all vertices, therefore any tree of H forms a complete family. Figure 4.4(b) shows the single tree $T = \{e_{s,t}, e_{s,d_1}, e_{d_1,d_2}\}$ that forms a complete family. Our algorithm will find a slightly larger complete family as we do not try to optimize the size of the complete family. However, for simplicity of the presentation, we use the complete family $\{T\}$. Note T has toughness 2.

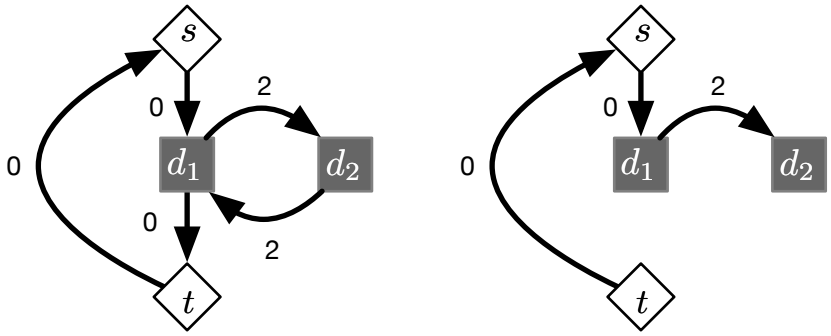


Figure 4.4. The condensed graph H and the tree T, which forms a generating family.

One can generate a generating family from the complete family $\{T\}$. There is a unique A-avoiding path from s to t and from s to d_1, so we only have to look at the path from d_1 to d_2. Here is the list of all possible trees that are topologically equivalent to T, which together form a generating family.

1. $T_1 = \{e_{s,t}, e_{s,d_1}, e_{d_1,s_1}, e_{s_1,r_2}, e_{r_2,d_2}\}$
2. $T_2 = \{e_{s,t}, e_{s,d_1}, e_{d_1,s_1}, e_{s_1,r_1}, e_{r_1,d_2}\}$
3. $T_3 = \{e_{s,t}, e_{s,d_1}, e_{d_1,s_1}, e_{s_1,d_2}\}$

4. $T_4 = \{e_{s,t}, e_{s,d_1}, e_{d_1,r_1}, e_{r_1,d_2}\}$
5. $T_5 = \{e_{s,t}, e_{s,d_1}, e_{d_1,r_2}, e_{r_2,d_2}\}$
6. $T_6 = \{e_{s,t}, e_{s,d_1}, e_{d_1,d_2}\}$

In general, we are looking at $O(n^{2k-2})$ different trees. Finally, we find the optimal tour that must use the edges in T_i for each i. This gives us different potential solutions, and our algorithm takes the smallest. The solution of HMATSP-kFVS that has to use edges in T_1 is show in Figure 4.3(b), which translates to the optimal solution of the original problem, shown in Figure 4.2c. Just for exposition, consider we replaced T_1 with T_2, we would obtain the solution in Figure 4.2d, which has a cost of 14, therefore will be discarded by the algorithm by being larger than optimal.

4.2 Arbitrary start and end Locations

Gharehgozli et al. (2017b) also consider the case where the start and end locations are arbitrary. In particular, let ℓ_s and ℓ_t be arbitrary locations and we define the vertices $s = \ell_s$ and $t = \ell_t$. We build the graph G in a similar way as in section 3.2 with the following steps specific to this case. See Figure 4.5 for an example of the problem.

1. For each $v \in R \cup D$, there is an edge $e_{s,v}$. This encodes the information that at the start position, the machine can move to a retrieval location, a depot or the end position (if there is no request). The cost of the edge is the distance from s to v.
2. For each $v \in S \cup D$, there is an edge $e_{v,t}$. This encodes the information that we move to the end location from a storage location or a depot. The cost of the edge is the distance from v to t.

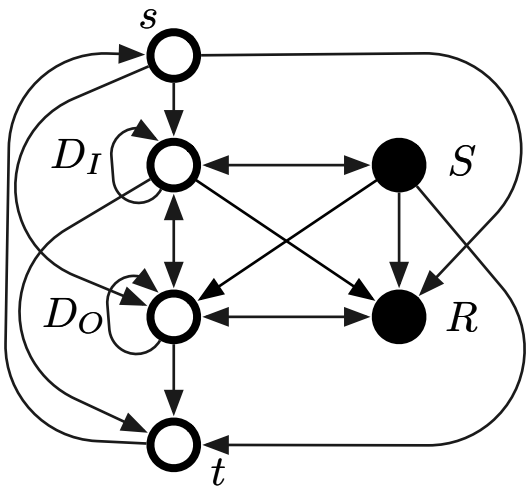


Figure 4.5. SRschedule(k) with arbitrary start and end locations

Theorem 4.3 SRSCHEDULE*(k) with k regular depots where arbitrary start and end locations are allowed can be solved in* $O(n^{2k+1})$ *time.*

Proof: Let the feedback vertex set be $A = D \cup \{s, t\}$ and H be the condensed graph of G. Note that in the condensed graph, $w(e_{s,u}) = w(e_{v,t}) = 1$, $w(e_{t,s}) = 0$, and all other edges have weight 2. Let H' be a strongly-connected subgraph of H. Note $e_{t,s}$ is an edge. We contract t and s into a single vertex. We find a spanning tree in H'. The spanning tree has weight at most $2k-1$. Uncontract s and t and it gives us a spanning tree in H' with weight at most $2k-1$. We obtain the theorem by applying Theorem 2.12. □

The above theorem extends the result of arbitrary start and end locations of Gharehgozli et al. (2017b) to k depots. In the case of two depots, we improve the computational complexity by a factor of n.

5 Numerical Experiments

In this section, we carry out several numerical experiments to evaluate the performance of our algorithm and generate managerial insights. We run numerical experiments for an AS/RS with 50 pallets in the X direction and 16 pallets in the Y direction. The systems also has 2 depots which are located at the bottom of the system. In the sensitivity analysis, we also consider AS/RSs with different sizes and numbers of depots. The number of storage and retrieval requests ranges from 10 to 50 requests. The percentage of retrieval requests is 50% of all requests, i.e., $\mathbb{P} = 50\%$. We also study systems with all storage requests to the ones with all retrieval requests. The locations of requests and corresponding depots are randomly generated by Monte Carlo simulation. All locations are uniformly distributed over the AS/RS rack. The S/R machine starts and ends its operations at the depots of the system. Other inputs can be found in Table 2. The parameters considered to run the numerical experiments are consistent with the literature (see, for example, Gharehgozli et al. 2017b, Yu and De Koster 2009a,b, 2012).

Table 2. Parameters used in the numerical experiments

Parameter	Range of scenarios	Fixed parameters
Number of requests (n)	$10, 20, 30, 40, 50$	
Number of depots (k)	$2, 3, 6, 9, 12$	
Percentage of retrievals ($\mathbb{P}$)	$0, 30, 60, 90, 100$	
Rack size	$10, 40, 50, 70, 100$ pallets in the X and $10, 40, 50, 70, 100$ pallets in the Y directions	
Storage strategy		Random
Pallet size in meter (width × depth × height)		Net: $1 \times 1.2 \times 1.2$ Gross: $1.2 \times 1.4 \times 1.5$
S/R machine		Vertical speed*: 24 meters/minute Horizontal speed: 80 meters/minute

* No acceleration or deceleration is considered. The time required for the depth moves or pick up and drop off a load is a constant, which can be omitted for optimization.

Algorithms discussed in Sections 3 and 4 are used to solve the problems to optimality in a few seconds. The results are shown in Table 3. In each scenario of the simulation study, a number of realizations of n requests with storage locations, storage input depots, and retrieval locations are randomly generated by Monte Carlo simulation. The number of realizations satisfies $N_{\text{realz.}} \geq \sigma^2\left(\frac{(1+\varepsilon)\mathbb{Z}_{1-\alpha/2}}{\varepsilon\mu}\right)^2$ with a 90% confidence level ($\alpha = 10\%$), where σ^2 and μ are respectively the variance and mean of the objective values, $\mathbb{Z}_{1-\alpha/2}$ is the $1-\alpha/2$ percentile of the normal distribution, and $\varepsilon = 5\%$ is the width of the confidence interval (Law and Kelton 1999). The study is performed on a Notebook with 2.11 GHz Intel® Core™ i7 processor, with 16.00 GB of RAM and the programming language is Python 3.8.The following observations can be made from this table.

- Observation 1. By increasing the number of requests, the total travel time soars. Furthermore, the total travel time is longer in larger AS/RSs, since the S/R machine needs to travel larger distances to retrieve and store requests. On the other hand, the number of depots does not have an impact on the total travel time. In the current era of e-commerce that online shopping is becoming increasingly popular, not only the number of requests but also the size of AS/RSs in distribution centers are constantly increasing. In addition, giant e-commerce retailers, such as Amazon, nowadays offer

overnight delivery, if products are ordered before a certain time. Therefore, the throughput time of orders in the fulfillment centers becomes an important element (every second matters) that has to be well managed using latest technologies (for example, Amazon Robotics, formerly known as Kiva Systems, Swisslog, Interlink, GreyOrange and Mobile Industrial Robots) and advanced techniques such as the ones developed in this paper.

- Observation 2. We compare our solution with random, the nearest neighbor(NN) and first come first served (FCFS) heuristics commonly used in practice. The optimal solutions outperform the ones obtained by the FCFS and NN heuristics by more than 30% and 20%, respectively. Today, a humongous number of orders need to be fulfilled fast, accurately, and inexpensively in distribution centers. Therefore, relying on heuristics, traditionally used in practice with such huge gaps, results in long lead times and unsatisfied customers.

- Observation 3. In instances with more storage or retrieval requests, the gap between the optimal solutions and the ones by the heuristics decreases. Furthermore, the total travel time of the S/R machine increases. In such instances, the S/R machine losses the opportunity to take advantage of double cycles. Operations mangers in distribution centers need to take this point into consideration when a set of requests are assigned to an S/R machine. Balanced combinations of storage and retrieval requests can result in shorter travel times.

- Observation 4. The results show that the algorithm is robust and the solutions can be obtained in less than a second, almost in all instances. For a fixed number of requests and depots, the computation time is quite insensitive to changes in problem inputs such as the size of the system and percentage of retrieval requests in different scenarios. However, by increasing the number of requests and depots, the computation time increases. Since the complexity of the algorithm, $O(n^{2k} + n^3)$, contains the $2k^{th}$ power of n (i.e., k and n are the number of depots and requests, respectively), the number of depots has a more dramatic impact on the computation time.

Table 3. Comparison of the optimal and heuristic solutions

X	Y	k	n	$\mathbb{P}$ (%)	Z^*	Z_{FCFS}	Z_{NN}	G_{FCFS} (%)	G_{NN} (%)	Ave. CPU
50	16	2	**10**	50	435.01	607.14	583.47	28.35	25.44	0.01
50	16	2	**20**	50	883.72	1337.89	1179.16	33.95	25.06	0.02
50	16	2	**30**	50	1263.78	1900.43	1610.46	33.50	21.53	0.03
50	16	2	**40**	50	1628.91	2513.18	2187.7	35.19	25.54	0.05
50	16	2	**50**	50	2051.99	3239.01	2566.93	36.65	20.06	0.08
10	16	2	20	50	748.29	1101.61	901.62	32.07	17.01	0.02
40	16	2	20	50	791.21	1182.41	1028.66	33.08	23.08	0.02
70	16	2	20	50	947.93	1426.15	1247.09	33.53	23.99	0.02
100	16	2	20	50	1203.26	1892.81	1667.23	36.43	27.83	0.02
50	**10**	2	20	50	679.37	1036.57	888.57	34.46	23.54	0.02
50	**40**	2	20	50	1843.86	2703.89	2223.99	31.81	17.09	0.02
50	**70**	2	20	50	2895.94	4396.72	3464.8	34.13	16.42	0.02
50	**100**	2	20	50	4290.87	6162.18	5220.32	30.37	17.80	0.02
50	16	**1**	20	50	839.45	1233.69	1052.99	31.96	20.28	0.01
50	16	**3**	20	50	867.72	1299.45	1173.04	33.22	26.03	0.05
50	16	**4**	20	50	846.1	1258.96	1136.56	32.79	25.56	9.54
50	16	2	20	**0**	1491.37	1599.45	1497.06	6.76	0.38	0.01
50	16	2	20	**30**	1061.76	1360.83	1265.9	21.98	16.13	0.02
50	16	2	20	**60**	901.4	1247.07	1054.49	27.72	14.52	0.02
50	16	2	20	**100**	1388.93	1417.52	1398.27	2.02	0.67	0.01

NOTE. Let Z^* be the average optimal solution obtained by our algorithm. The heuristics are quick and can find the solution in less than a second. Z_{FCFS} and Z_{NN} are the average objective values (total travel times) obtained by the FCFS and NN heuristics. Furthermore, we calculate the gaps as $G_{\text{FCFS}}(\%) = (Z_{\text{FCFS}} - Z^*)/(Z_{\text{FCFS}}) \times 100$, and $G_{\text{NN}}(\%) = (Z_{\text{NN}} - Z^*)/(Z_{\text{NN}}) \times 100$. Finally, the Ave. CPU column shows the average computation times (in seconds).

In the remaining of this section, we compare our algorithm with the ones in the literature. As discussed in Table 1, Gharehgozli et al. (2017b) have developed an $O(n^6)$ step algorithm for an AS/RS with 2 depots. This algorithm and the one developed in this paper obtain the same optimal solutions. However, our algorithm outperforms theirs in terms of computation time. To quickly find an optimal solution, they use a subtour merging procedure to merge the subtours of an AP relaxation optimal solution, using the requests which need to be retrieved to or stored from the same depot. As shown in Table 4, without this procedure, the computation time can peak to more than 1,800 seconds for an instance with 50 requests. A similar procedure has been also implemented in our algorithm. However, by canceling this procedure, the computation times still remain reasonable. Our algorithm has an $O(n^5)$ complexity which is n times smaller than their algorithm. This becomes critical in instances that the subtour merging procedure cannot be executed. In such instances, our algorithm can quickly find a solution whereas the other algorithm can take a long time, which is not acceptable in large e-commerce centers processing a huge number of orders. Finally, for a setting with k depots, Gharehgozli et al. (2014) have developed a branch & bound algorithm (see Table 1). To run numerical experiments, they use the context of a container terminal with different input parameters. Furthermore, they do not report their computation times without the subtour merging procedure. Therefore, we are not able to compare our results with theirs. However, similar conclusions are expected for the comparison of our algorithm with theirs as well. Our algorithm runs in polynomial time and can consistently find an optimal solution in $O(n^{2k}+n^3)$ steps, which outperforms a branch & bound algorithm.

Table 4. Comparing our Computation times with the ones by Gharehgozli et al. (2017b)

X	Y	k	n	$\mathbb{P}$ (%)	Z^*	Z_{FCFS}	Z_{NN}	G_{FCFS} (%)	G_{NN} (%)	Ave. CPU^1	Ave. CPU^2
50	16	2	**10**	50	449.06	629.3	544.69	28.64	17.56	0.61	0.15
50	16	2	**20**	50	820.91	1236.99	1048.19	33.64	21.68	20.27	0.96
50	16	2	**30**	50	1240.39	1903.72	1609.64	34.84	22.94	187.04	4.19
50	16	2	**40**	50	1633.73	2487.12	2146.66	34.31	23.89	776.24	11.31
50	16	2	**50**	50	2005.75	3187.65	2603.5	37.08	22.96	1811.57	23.21

NOTE. Let Ave. CPU^1 and Ave. CPU^2 columns report the computation times by Gharehgozli et al. (2017b) and by our algorithm, respectively. The average optimal and heuristic objective values are reported by Gharehgozli et al. (2017b).

6 Conclusion

Automated storage/retrieval systems (AS/RSs) have received an increasing attention in academia and practice, due to their faster, cheaper and more efficient performance. The need for shorter lead times in the era of e-commerce has made AS/RSs even more popular. We study the problem of sequencing storage and retrieval requests to be carried out by an S/R machine in an AS/RS with k depots. We model the problem as a variation of the traveling salesman problem, the high multiplicity asymmetric traveling salesman problem with feedback vertex set and propose a polynomial time algorithm to solve the problem as long as the size of the feedback vertex set is constant. The algorithm improves upon previous best known results on the two depot instances and extends the results to k depot instances as well. Applications to variations of the sequencing problem are discussed and computational complexities are presented. The numerical experiments show that our solution method can provide better solutions as compared with the ones obtained by the heuristics commonly used in practice. Furthermore, increasing the number of requests and the size of the system results in a longer total travel time. On the other hand, the number of depots does not impact the total travel time. Finally, a balance between the number of storage requests and retrieval requests assigned to an S/R machine decreases the total travel time, since more double cycles can be performed by the machine.

Acknowledgments The authors would like to thank the editor and anonymous reviewers whose constructive comments and suggestions helped us to improve the quality of this paper.

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

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

Bang-Jensen, J., G. Z. Gutin. 2008. *Digraphs: Theory, Algorithms and Applications (Springer Monographs in Mathematics)*. Springer.

Berger, André, László Kozma, Matthias Mnich, Roland Vincze. 2019. A time- and space-optimal algorithm for the many-visits TSP. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. 1770–1782.

Boysen, N., K. Stephan. 2016. A survey on single crane scheduling in automated storage/retrieval systems. *European Journal of Operational Research* **254**(3) 691 – 704.

Boysen, Nils, R. de Koster, Felix Weidinger. 2019. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research* **277**(2) 396 – 411.

Burkard, R. E., V. G. Deineko, R. Van Dal, J. A. A. Van der Veen, G. J. Woeginger. 1998. Well-solvable special cases of the traveling salesman problem: A survey. *SIAM Review* **40**(3) 496–546.

Carlo, H. J., I. F. A. Vis, K. J. Roodbergen. 2014. Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research* **235**(2) 412 – 430.

Cayley, A. 1889. A theorem on trees. *Quart. J. Math.* **23** 376–378.

Cosmadakis, S., C. Papadimitriou. 1984. The traveling salesman problem with many visits to few cities. *SIAM Journal on Computing* **13**(1) 99–108.

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

Gharehgozli, A. H., G. Laporte, Y. Yu, R. de Koster. 2015. Scheduling twin yard cranes in a container block. *Transportation Science* **49**(3) 686–705.

Gharehgozli, A. H., D. Roy, R. De Koster. 2016. Sea container terminals: New technologies and or models. *Maritime Economics & Logistics* **18**(2) 103 – 140.

Gharehgozli, A. H., F. G. Vernooij, N. Zaerpour. 2017a. A simulation study of the performance of twin automated stacking cranes at a seaport container terminal. *European Journal of Operational Research* **261**(1) 108 – 128.

Gharehgozli, A. H., Y. Yu, R. De Koster, J. T. Udding. 2014. An exact method for scheduling a yard crane. *European Journal of Operational Research* **235**(2) 431 – 447.

Gharehgozli, A. H., Y. Yu, X. Zhang, R. de Koster. 2017b. Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system. *Transportation Science* **51**(1) 19–33.

Gharehgozli, Amir, Yugang Yu, René B.M. de Koster, Shaofu Du. 2019a. Sequencing storage and retrieval requests in a container block with multiple open locations. *Transportation Research Part E: Logistics and Transportation Review* **In press**.

Gharehgozli, Amir, Nima Zaerpour. 2018. Stacking outbound barge containers in an automated deep-sea terminal. *European Journal of Operational Research* **267**(3) 977 – 995.

Gharehgozli, Amir, Nima Zaerpour, Rene de Koster. 2019b. Container terminal layout design: transition and future. *Maritime Economics & Logistics* 1–30.

Gorman, M. F., J. P. Clarke, A. H. Gharehgozli, M. Hewitt, R. De Koster, D. Roy. 2014. State of the practice: A review of the application of or/ms in freight transportation. *Interfaces* **44**(6) 535–554.

Grigoriev, Alexander, Joris Van De Klundert. 2006. On the high multiplicity traveling salesman problem. *Discrete optimization* **3**(1) 50–62.

Karp, R. M. 1972. Reducibility Among Combinatorial Problems. R. E. Miller, J. W. Thatcher, eds., *Complexity of Computer Computations*. Plenum Press, 85–103.

Kim, B. I., S. S. Heragu, R. J. Graves, A. S. Onge. 2003. Clustering-based order-picking sequence algorithm for an automated warehouse. *International Journal of Production Research* **41**(15) 3445–3460.

Law, A. M., D. M. Kelton. 1999. *Simulation Modeling and Analysis*. 3rd ed. McGraw-Hill Higher Education, Boston, Massachusetts.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. 1985. *The Traveling Salesman Problem, a Guided Tour of Combinatorial Optimization*. John Wiley & Sons, Chichester, UK.

Lee, H. F., S. K. Schaefer. 1997. Sequencing methods for automated storage and retrieval systems with dedicated storage. *Computers & Industrial Engineering* **32**(2) 351 – 362.

Li, W., M. Goh, Y. Wu, M. E. H. Petering, R. De Souza, Y. C. Wu. 2012. A continuous time model for multiple yard crane scheduling with last minute job arrivals. *International Journal of Production Economics* **136**(2) 332–343.

Li, W., Y. Wu, M. E. H. Petering, M. Goh, R. de Souza. 2009. Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research* **198**(1) 165–172.

Man, X., F. Zheng, F. Chu, M. Liu, Y. Xu. 2019. Bi-objective optimization for a two-depot automated storage/retrieval system. *Annals of Operations Research* .

Psaraftis, Harilaos N. 1980. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research* **28**(6) 1347–1359.

Ramtin, Faraz, Jennifer A. Pazour. 2014. Analytical models for an automated storage and retrieval system with multiple in-the-aisle pick positions. *IIE Transactions* **46**(9) 968–986.

Ramtin, Faraz, Jennifer A. Pazour. 2015. Product allocation problem for an as/rs with multiple in-the-aisle pick positions. *IIE Transactions* **47**(12) 1379–1396.

Roodbergen, K. J., I. F.A. Vis. 2009. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research* **194**(2) 343 – 362.

Sarin, Subhash C., Hanif D. Sherali, Liming Yao. 2011. New formulation for the high multiplicity asymmetric traveling salesman problem with application to the chesapeake problem. *Optimization Letters* **5**(2) 259–272.

Schwerdfeger, Stefan, Nils Boysen. 2017. Order picking along a crane-supplied pick face: The sku switching problem. *European Journal of Operational Research* **260**(2) 534 – 545.

Tappia, Elena, Debjit Roy, Marco Melacini, René De Koster. 2019. Integrated storage-order picking systems: Technology, performance models, and design insights. *European Journal of Operational Research* **274**(3) 947 – 965.

Van den Berg, J. P., A. J. R. M. Gademann. 1999. Optimal routing in an automated storage/retrieval system with dedicated storage. *IIE Transactions* **31**(5) 407.

Vis, I. F. A., K. J. Roodbergen. 2009. Scheduling of container storage and retrieval. *Operations Research* **57**(2) 456–467.

Yu, H., Y. Yu. 2019. Optimising two dwell point policies for as/rss with input and output point at opposite ends of the aisle. *International Journal of Production Research* **0**(0) 1–19.

Yu, Y., M. B. M. De Koster. 2009a. Designing an optimal turnover-based storage rack for a 3D compact automated storage and retrieval system. *International Journal of Production Research* **47**(6) 1551–1571.

Yu, Y., R. B. M. De Koster. 2009b. Optimal zone boundaries for two-class-based compact three-dimensional automated storage and retrieval systems. *IIE Transactions* **41**(3) 194–208.

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

Zaerpour, Nima, René B.M. de Koster, Yugang Yu. 2013. Storage policies and optimal shape of a storage system. *International Journal of Production Research* **51**(23-24) 6891–6899.

Zaerpour, Nima, Rosalie Volbeda, Amir Gharehgozli. 2019. Automated or manual storage systems: do throughput and storage capacity matter? *INFOR: Information Systems and Operational Research* **0**(0) 1–18.

Zaerpour, Nima, Yugang Yu, R. B. M. de Koster. 2015. Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Production and Operations Management* **24**(8) 1266–1284.