FORKLIFT ROUTING OPTIMIZATION IN A WAREHOUSE USING A CLUSTERING-BASED APPROACH

Saif Muhammad Musfir Rahman

A thesis
in
The Department
of
Department of Mechanical, Industrial & Aerospace Engineering
(MIAE)

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science in Industrial Engineering
Concordia University
Montréal, Québec, Canada

May 2019

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Saif Muhammad Musfir Rahman**

Entitled: **Forklift Routing Optimization in a Warehouse using a Clustering-based Approach**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Industrial Engineering

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Ida Karimfazli _____ Chair

Dr. Mingyuan Chen _____ Internal Examiner

Dr. Farnoosh Naderkhani _____ External Examiner

Dr. Daria Terekhov _____ Supervisor

Dr. Satyaveer S. Chauhan _____ Co-supervisor

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____

Amir Asif, Ph.D.,ing., FEIC, FCSME, FASME, Interim Dean
Faculty of Engineering and Computer Science

Abstract

Forklift Routing Optimization in a Warehouse using a Clustering-based Approach

Saif Muhammad Musfir Rahman

Order picking in a warehouse is considered to be a time-consuming and costly process that results in loss of profit for the management. Hence a warehouse management team is always looking to improve their picking process and increase their efficiency. In this research, a warehouse with narrow aisles is studied. The aisles are so narrow that a forklift is only allowed to traverse them in one direction thus making them unidirectional. The picking process is modeled first as an uncapacitated vehicle routing problem and then as a capacitated vehicle routing problem. The objective is to minimize the total travel distance. Since the Mixed Integer Programming model takes a long time to solve large instances, we develop a heuristic algorithm both for the uncapacitated and capacitated problems by combining two methodologies of heuristics and machine learning. The algorithm is able to solve the instances to near optimality quickly, finding practical solutions that could potentially be implemented into actual warehouses to reduce order picking time and hence, overall warehouse costs.

Acknowledgements

I would first like to express my earnest thanks to my academic supervisors Dr. Daria Terekhov and Dr. Satyaveer Chauhan for their constant support, guidance and motivation. It would never have been possible for me to take this work to completion without their incredible support and encouragement. I would also like to thank my parents and brother, without their constant support and affection this would never be possible. Lastly, my appreciation goes to my friends and labmates who helped me with their knowledge and advice in every step of the journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main goal of this thesis is to develop a heuristic algorithm for a warehouse vehicle routing problem. The algorithm solves a warehouse forklift routing problem for large instances where it is too time consuming to be solved by a Mixed Integer Programming model. Order picking creates great concern for warehouse managers and routing imposes the biggest challenge in terms of time consumption. In order to solve the forklift routing problem of the warehouse of our industry partner, we develop a heuristic algorithm that generates a near-optimal solution quickly. In particular, in this thesis we

- Formulate a Mixed Integer Programming (MIP) model for two versions of a warehouse routing problem with unidirectional aisles, i.e., one where the forklift is uncapacitated and one where it has capacity,
- Develop a heuristic algorithm for both the uncapacitated and capacitated variants of the warehouse routing problem.

In order to develop the heuristic, we combine two methodologies: heuristics and unsupervised clustering. In particular, we use

- The S-shaped heuristic developed by Roodbergen [46]
- A variation of the K-means clustering algorithm [18].

The combination of these two algorithms is used to develop a heuristic that results in a near-to-optimal solution in smaller processing time compared to the MIP model. The contributions of this thesis therefore lie at the intersection of unsupervised clustering

and operations research. Only a handful of papers have addressed the problem of vehicle routing through clustering, as we will see in Section 2 (Literature Review).

1.1 Motivations

De Koster et al. [16] defined order picking as "a warehousing operation that deals with picking products from storage locations in order to satisfy customer orders". Order picking is one of the most challenging and expensive processes in a distribution center. It has been estimated that order picking accounts for nearly half of the total warehouse operating costs [19]. This situation is mostly due to the fact that order picking often requires involvement of human order pickers, as automating the process would be costly [16]. Hence, order picking has in recent years become an area of increased interest among warehouse professionals for improving productivity in warehouses [19].

This dissertation is completed in partnership with a company based in Québec similar to the work done by Chabot et al. [11] which operates in the warehouse, logistics and delivery service industry and possesses a large number of SKUs and large number of orders, each having few items. The products are stored in both sides of narrow aisles, as depicted in Figure 1.

Figure 1: General overview of a distribution center with narrow aisles [11]

As mentioned by Chabot et al., narrow aisles are advantageous to maximize floor space utilization, and they facilitate the picking process as the picker can pick items from shelves on both sides of the aisles directly [11, 23]. However, special narrow aisle forklift trucks are needed which are required to be driven according to some specific rules due to safety reasons [11]. In narrow aisle warehouses, traffic is often an important issue, but similar to Chabot er al. [11] in our case traffic problems are

eliminated since only one forklift is allowed to enter an aisle at a given time and the aisles have pre-defined directions through which the forklift can traverse them. Warehouse control comprises numerous challenges, namely operating, receiving, storing, order picking and shipping. Some of the most fundamental issues in order picking are batch processing and routing methods [16]. The constraints and specifications discussed above lead to challenges in terms of designing an efficient routing strategy, a problem that we address in this thesis.

In particular, the motivation for the work presented in this thesis is the need for effective methods for solving vehicle routing problems in large warehouses in short runtime. Order picking is an arduous task in warehouse processes that may take up more than half of all labor time of a warehouse [51]. These order pricking problems are modeled in the literature as a Vehicle Routing Problems (VRPs) which are solved by Mixed Integer Programming [38]. The problem with VRP MIP models lie in the fact that they may take a long time to find an optimal solution. When the number of products increases, solution time of the MIP model grows exponentially which is highly undesirable by warehouse management. The main contribution of this thesis is that we propose a heuristic algorithm that allows us to deal with a large size warehouse VRP and to solve this problem in a short time using combination of a heuristic method and a clustering method.

1.2 Outline

In Chapter 2 of this thesis, the necessary background for the work presented in this thesis is provided. The chapter commences by discussing warehouse order picking problems in general and then goes on to focus on warehouse routing solving techniques. This section includes four main subsections: Order picking problems, Warehouse routing solving techniques, Clustering and finally, Conclusion. Chapter 3 provides a specialized mixed integer programming mathematical model for the vehicle routing problem of the warehouse in concern. In this chapter we assume that the forklift does not have any capacity, i.e., is uncapacitated. We develop the heuristic for the uncapacitated case and discuss the results obtained from our experiments. In Chapter 4, the mixed integer programming model and the heuristic algorithm for the

capacitated case are developed. We no longer assume that a forklift is uncapacitated, rather it has a finite capacity which is more aligned to a realistic scenario. We run experiments for our model through several capacitated scenarios and discuss the results obtained from our experiments. Chapter 5 concludes this thesis by re-stating its main contributions and suggesting some areas for future work.

1.3 Summary of Contributions

The core contribution of this thesis is that it designs and develops a heuristic to solve the vehicle routing problem in a warehouse with a large number of unidirectional aisles where completing each route in the shortest time is paramount. To do so, this thesis combines a heuristic approach with a clustering technique, thus contributing in the combined domain of operations research and unsupervised clustering. First, we formulate specialized MIP models for the warehouse routing problem with unidirectional aisles both for uncapacitated and capacitated cases. Second, we develop heuristic algorithms for both cases. Third, we show that the developed heuristic, using unsupervised clustering technique, provides near-to-optimal solutions in very short time for large instances.

Chapter 2

Literature review

This chapter provides an overview of the main topics and the most relevant studies in order to identify and highlight the recent state of the art methods in the research areas relevant to the problem studied in this thesis. This chapter is divided into four different sections:

(1) Section 2.1 - Order picking problems
(2) Section 2.2 - Warehouse routing solving techniques
(3) Section 2.3 - Clustering
(4) Section 2.4 - Conclusion

2.1 Order Picking Problems

Order picking is a source of headache for warehouse operations management still since it often requires involvement of human order pickers, as automating order picking systems requires large investments [16]. Hence, order picking has become an area of increased interest among warehouse and logistics professionals to increase productivity in warehouses [16]. The term "warehouse" refers to "a facility where the main purpose is storage and buffering of products" [47]. In such facilities products are not stored for a long time and are transferred directly from receiving to shipping dock [32]. Warehouse operations can be divided into several functions [32], but in this thesis, we are concerned with the order picking problem of the warehouse. According to

Dallari et al. [14], the design of order picking systems depends on several warehousing elements, ranging from products (e.g. number, size, value), customer orders (e.g. number, size), to design and layout of warehouse areas. In this thesis, our problem falls under the "picker-to-parts" order picking system which Dallari et al. [14] defined as a system which involves human pickers moving along the aisles picking products on foot using carts or riding on specialized vehicles. Pickers can pick a single order at a time or multiple orders in a batch [14]. Order picking optimization focuses on minimizing the total order picking time [57]. The time of a single picking tour is composed of travel time, search time, pick time and set-up time [17].

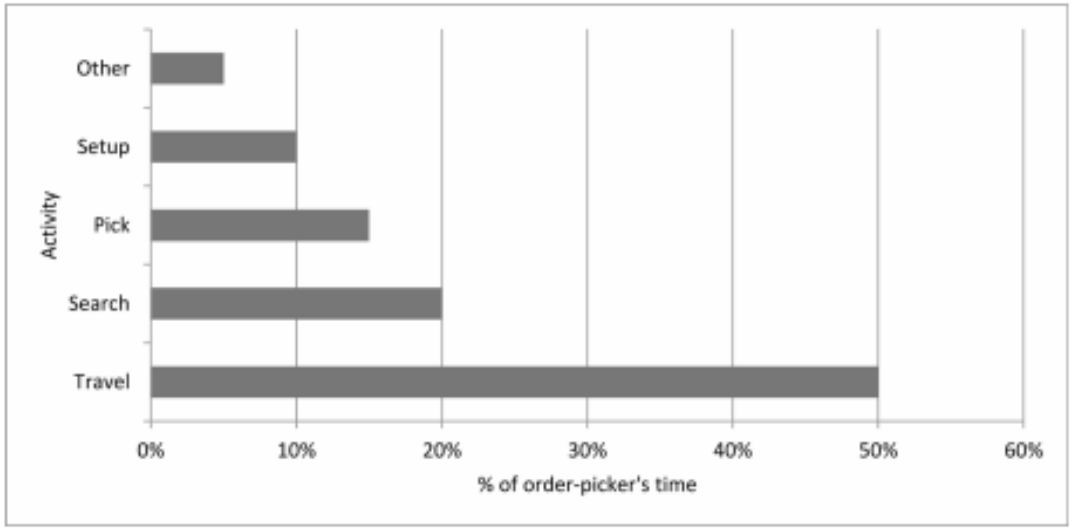


Figure 2: Typical order picker's time distribution

Figure 2 shows a typical order picker's time distribution on average as studied by Tompkins et al. [52]. Traveling is the most time consuming subtask of order picking. Pick and search times are often constant irrespective of picking sequence while set-up time is most often ignored [27]. Hence, we are only left with travel time as the objective. Moreover, assuming that order pickers travel in constant speed results in optimization of total travel distance [27, 21].

Order picking problem with the objective of minimizing the total distance has been studied extensively in literature in different warehouse layouts. Chabot et al. [11] looked at minimizing the travel time to collect all ordered items distributed in a warehouse with 3D narrow aisles. In 3D aisles, different products can be placed in

front of each other, giving the notion of "depth" for each slot or shelf. The problem that we consider is a simpler version of this, where each slot corresponds to one kind of pallet or product. Lu et al. [36] look at a warehouse routing problem with narrow aisles but can be traversed in both directions. One of the characteristics of our problem, however, is that the aisles are unidirectional. Mohr [38] studied a warehouse with multiple zones where the picking policy is manual picker-to-part and pick by order. The problem that this thesis tackles is also similar to the problem addressed in this paper in terms of the picking policy being "picker-to-part". Hassan and Ferrell [24] introduced the idea of a stackability matrix where the items can be stacked on top of another during transport and the objective is to determine time optimal routes for the picker in a manual warehouse. We, however, do not consider any stackability matrix for our problem. Roodbergen et al. [46] considered routing and layout issues for parallel aisle warehouses with multiple cross aisles. A cross aisle is defined as "an aisle which is perpendicular to the storage aisles in which the products are stored. Its main function is to enable aisle changing" [55]. However, our warehouse does not contain any cross aisle. Despite these special cases, most order picking problems in the real world are much more complex as they need to handle multiple operation specific constraints. One of theses constraints is the presence of narrow aisles. Narrow aisles allow the maximum utilization of floor space and allow the picker to pick products from both sides of the aisles [11, 23]. The literature of warehouse routing can be vast and varied since each warehouse can have specific constraints that are distinct to that warehouse.

2.2 Warehouse Routing Solving Techniques

Routing policy is concerned with ordering a list of items to be picked that will minimize the travel distance of an order picker [29]. The optimization problem is a variant of the well-known Traveling Salesman Problem (TSP) [44]. Such cases are usually solved using meta heuristic algorithms which provide suboptimal solutions. However, for some warehouse configurations, optimal polynomial-time algorithms do exist, as proposed by Ratliff and Rosenthal [44].

2.2.1 Polynomial-time heuristics

The optimal algorithm presented in Ratliff and Rosenthal [44] is complex and not easily adaptable to different warehouse layouts [17]. Moreover, the sequences produced with such algorithms are often not straight-forward and seem illogical to order pickers, who then as a result deviate from the routes calculated [17, 21], thus defeating the purpose optimizing picking tours. However, to tackle these issues, standardized routing with various routing strategies is often used in practice [57]. As discussed in [29], such routing strategies include:

- S-shaped strategy: The order picker traverses an entire aisle containing at least one item. This results in the order picker changing between entering aisles from front and back cross aisle.

- Return strategy: The order picker visits an aisle depending on whether there are pick locations in that aisle or not. He/She enters it from the front cross aisle and picks the products from the order list in that aisle. After picking the farthest placed product, he/she returns to the front cross aisle. The return strategy requires the aisles to be bi-directional.

- Largest Gap strategy: This strategy tries to maximize the distance in the aisle that is not traversed. A gap is the distance between the start of an aisle and the first location where an item is located, two adjacent pick locations, and last pick location and end of the aisle. The order picker returns to the start of the aisle at the pick location that is part of the largest gap.

The three routing strategies above are depicted in Figure 3. Petersen [43] compared six routing strategies including return, S-shaped and largest gap for different warehouse layouts [29]. The results as also discussed in [38] showed *largest gap* and *composite* were the best routing strategies that were closest to giving an optimal solution. However from the above routing strategies, it is seen that both *return* and *largest gap* require the aisles to be bi-directional. Hence, if the aisles are unidirectional, the $S - shaped$ routing strategy needs to be implemented.

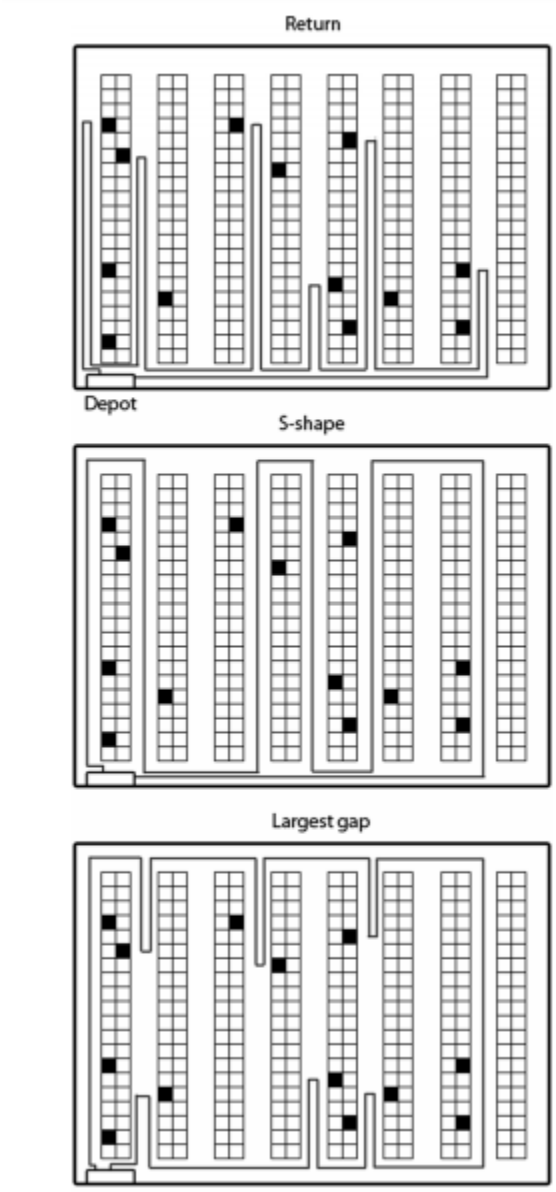


Figure 3: Different order picking routing strategies [29]

2.2.2 Mixed Integer Programming Formulations

The MIP formulations used to solve warehouse routing problems are based on classical formulations, in particular the Traveling Salesman Problem, the Directed Rural Postman Problem and the Vehicle Routing Problem.

Traveling Salesman Problem

The pursuit of finding an optimal or near-optimal order picking route of an order-picker problem in a typical rectangular warehouse with multiple aisles is classified as the Steiner Traveling Salesman Problem (STSP) [16, 50]. There are two general methods to solve the STSP [50]: the first one is to transform a STSP into the classic TSP by computing the shortest paths between every pair of nodes [16] (this is the methodology applied in this thesis) and the other is to apply algorithms such as the polynomial-time heuristic algorithms discussed above.

Goetschalckx and Ratliff [22] proved that the optimal aisle traversal problem can be modeled and solved as a shortest path problem. Roodbergen and de Koster [46] developed an algorithm to find the shortest picking tour in a parallel aisle warehouse with a middle aisle. An exact approach using dynamic programming had been proposed for the first time by Ratliff and Rosenthal in the case of a single cross-aisle [44]. Cambazard and Catusse developed a dynamic programming approach which is able to solve any rectilinear TSP [10], but the algorithm is exponential in number of horizontal lines.

Vehicle Routing Problem

Laporte et al. [34], however, used a VRP model to solve a family of multi-depot vehicle routing and location-routing problems. The delivery and pick-up problem is a generalization of the VRP, which is a generalization of the traveling salesman problem (TSP), the well-known hard combinatorial optimization problem [41]. Anbuudayasankar and Mohandas [2] developed a mixed-integer programming model for VRP with simultaneous delivery and pick-up with an additional constraint of maximum route-length.

The capacitated vehicle routing problem (CVRP) is more similar to the problem tackled in this thesis since it requires the truck to return to the starting point after having visited a subset of nodes. More recently Chabot et al. [11] described the Capacitated Narrow Aisle Order Picking Problem (CNA-OPP) and showed how to

model it as a VRP. Their work is very similar to the problem addressed in this thesis. However, the paper does not address the issue of uni-directional aisles.

The formulation followed in this paper is the VRP, hence we introduce a short literature review for the VRP model in warehouse scenario.

The VRP was originally proposed by Dantzig and Ramser in 1959 [15]. There is a significant amount of literature regarding the VRP. The standard definition of VRP is described by Laporte [33] as given in [38]: Let $G = (V, A)$, where V is a set of vertices, and A is a set of arcs representing the path from one node to another. Distance, cost or time is typically represented by $c_{i,j}$. The main objective of the VRP is to minimize the total distance or time traveled by a set of vehicles while respecting a set of constraints. VRP constraints are typically of the following nature:

- Each location represented by a vertex, apart from the depot, must be visited at least once.
- Each trip by a vehicle starts and ends at the depot.
- A set of additional constraints specific to the particular problem.

A comprehensive summary of several basic VRP and their formulations is provided by Toth and Vigo (2002) [53] which have been summarized further by Mohr [38] as follows:

- Capacitated VRP (CVRP): This problem is a VRP in which every vehicle has a capacity. The customers usually represent deliveries or pickups and the demands are deterministic. If split deliveries are not permitted (as in this thesis), then the entire demand must be met by one vehicle in order to be eligible to visit that node.
- VRP with Time Windows (VRPTW): Time windows represent the span of time during which the order nodes should be visited. The service time at each customer along with the depot exit time for the vehicle and its traveling time to the customer is tracked. Early arrival to the customer is allowed, but the vehicle has to wait before the beginning of its service time.

- VRP with backhauls: The problem is structured such that customers are divided into two parts: those that are visited on the way out from the depot and those that are visited on the way back to the depot. A constraint is imposed such that all the outbound customers are visited before visiting any of the inbound customers.

- VRP with Pickup and Delivery: This type of VRP consists of delivery demands and pickup quantities at each customer or location.

Directed Rural Postman Problem

The problem of unidirectional aisles was studied by Benavent and Soler [5] when they introduced a generalization of the Directed Rural Postman Problem (DRPP) for turns that are forbidden and other turns are allowed but with penalties. More recently, Colombi and Mansini [12] studied the known mathematical DRPP formulation and added valid inequalities and developed a branch and cut algorithm.

Summary and Justification of the Chosen Representation

As mentioned above, the formulation that we opt for in our problem is the VRP. This is because TSP requires the vehicle to return to the starting point (depot) after visiting all the nodes but it only allows the vehicle to make a single return. Hence, we cannot incorporate the capacity constraint of the problem in capacitated case. The Uncapacitated case though can be modeled as a TSP. The VRP, however, is a multiple-route node-service-combination problem with vehicle capacity limitation that allows the vehicle to make multiple tours with each tour visiting different set of nodes, which is exactly what we want in our solution.

The DRPP is a specialized form of TSP visiting each route between nodes at least once while returning to the origin and taking the shortest route among all possible routes that fulfill this criteria. It requires all edges of a given set of required edges be visited, not that all vertices be visited. In other words, DRPP tries to find an Eulerian cycle rather the Hamiltonian cycle. But, the problem in our case focuses on

the nodes rather than the edges connecting the nodes. We model the location of the items as nodes and therefore our focus is on visiting nodes, and not on using specific arcs. At first glance, it looks like the model should have a node for every location of every item. This is the approach that was chosen by Bant et al. in [48]. However, given the size of the warehouse as they found in their work, doing so is going to lead to a prohibitively large model. Second, we could consider nodes as the start and end of each aisle and that as long as we visit an edge, we have picked up all the items on that edge — this will give us the DRPP. Although this DRPP would be able to solve the uncapacitated case, for the capacitated case again, we are unable to incorporate the constraint of maximum capacity and hence resort to VRP as choice of our model. Finally, a modeling approach that overcomes both of the above issues is to model the locations of items as nodes, and furthermore to have the edges represent the shortest path between those two locations and solve it by VRP model. The graphical representation of our problem is discussed in detail in Chapters 3 and 4. We take motivation from the vehicle flow model of the VRP discussed in the work done by Toth and Vigo [53] and the VRP model introduced by Achutan and Caccetta [1].

2.2.3 Meta-Heuristic Methods

For large scale instances of any of the formulations above, it is nearly impossible to solve problems to optimality in reasonable amount of time using exact methods. Hence researchers turn to meta heuristics that search the solution space much quicker and more effectively often resulting in near optimal solutions. Vidal et al. [56] provide a comprehensive overview of many meta heuristics developed throughout the literature and comparison of their performances for various types of Vehicle Routing Problem [38]. The meta heuristics were categorized as neighborhood searches, population-based methods, hybrid and parallel or cooperative [38].

In the neighborhood category there is Simulated Annealing and Tabu Search [38]. Brandao and Mercer [8] uses a tabu search heuristic for a multi-trip VRP with time windows and a vehicle fleet that is heterogeneous in terms of capacity [38]. Chabot et al. [11] developed an Adaptive Large Neighborhood Search heuristic to compare the solution to their CVRP model of the 3D narrow aisle warehouse. Jin et al. [30] applies a tabu search algorithm for a classical CVRP that is based on the use of several

different neighborhood structures and performs parallel search [38]. Nguyen et al. [42] use tabu search to solve a VRP with time windows in a warehouse featuring one depot and multiple zones. Wang et al. [20] proposed a Genetic-Ant Colony algorithm that had good overall search ability for a VRP.

With population-based category, there is Genetic Algorithms (GAs). Mohr [38] applied and compared genetic algorithm to a capacitated vehicle routing problem in a warehouse with multiple zones. Baker and Ayechew [4] use genetic algorithm to solve a basic CVRP considering one depot and deterministic demand but no constraint in the direction of aisle traversal. Berger and Barkaoui [6] develops a parallel hybrid GA to solve a VRP where time windows are present [38]. Ursani et al. develop a novel GA which they separately applied to various independent locations. They called this Localized Genetic Algorithm and applied it to small scale CVRP [38]. We take motivation from this paper that a problem can be subdivided into smaller parts and solved independently. The initial population is generated using a nearest neighbor algorithm. The next customer visited for each vehicle is assigned based on its current nearest customer [38]. We adopt a similar idea to our model based on clustering.

2.3 Clustering

The method of dividing a problem into smaller problems and then solving each independently has already been applied in literature. Clustering adopts a similar approach and with the advent of machine learning techniques, this field is of particular interest. The vehicle routing problem in a warehouse is a pickup problem and one of the most reasonable improvements for pick up routing optimization is the idea of partitioning a graph into a subgraph, where a subgraph contains a limited number of untraversed vertices [49, 31].

The practical application of subgraph partitioning to the pick path problem lies in the fact that there is no need to take into consideration an area of a warehouse if no order locations are present in that side of the warehouse. The drawback with the subgraph concept, however, is that there must be an algorithm run to determine what subgraph should be looked at and then construct that subgraph [39, 31].

Another algorithm to explore that has similar attributes to clustering is Nearest Neighbor. The Nearest neighbor as discussed in [31] starts at a random point closest to the current point, until all the points have been visited. This algorithm, however, cannot guarantee any degree of accuracy as to how close it will be to the optimum solution [35]. Hence, clustering is considered to be a better alternative as is also seen in the literature. Clustering can be defined as "an unsupervised learning that divides data into groups or clusters. Cluster analysis is based on the principle of maximizing similarity within groups and minimizing between-group similarity" [9]. Clustering has several applications, mostly in machine learning, but it has also been used in operations research and routing problems specifically, as shown below.

One method to solve VRP is through hierarchical methods for which different methods can be applied [9]. Cluster first and then solve the routing problem or tackle the routing problem first and then the clustering can be given as examples of the hierarchical methods [9]. Dondo and Cerdã [18] used a cluster based optimization approach to solve a multi depot VRPTW with heterogeneous fleet. Hiquebran et al. [28] used simulated annealing algorithm based on a "cluster first, route second" approach to solve a VRP. Crainic et al. [13] applied a clustering based heuristic algorithm for two echelon VRP.

Boyzer et al. [7] developed a heuristic algorithm for the VRP where they first solved the clustering problem and then the routing problem. For the clustering stage, they used a fuzzy C-Means method and then used tabu search to improve the routing routes in the second stage [9]. Moolman et al. [40] implemented the DBSCAN to incorporate the predictive sharing of a resource which is used to solve the VRP [9]. He et al. [26] presented a balanced K-Means algorithm for partitioning areas in large scale vehicle routing problem. The traditional K-means was used to partition customers into several areas in the first stage and a border adjustment algorithm aims to adjust the unbalanced areas to be balanced in the second stage. Reed et al. [45] demonstrated the use of Ant Colony System to solve the capacitated vehicle routing problem associated with collection of recycling waste from households. K-Means clustering greatly improves the efficiency of the solution in networks where the nodes

are concentrated in separate clusters. More recently, Cömert et al. [9] developed a two stage hierarchical approach to solve a vehicle routing problem with time windows where they use three different clustering algorithms (i.e., K-Means, K-Medoids and DBSCAN) in the first stage and in the second stage, MILP is used to solve a VRPTW.

2.4 Conclusion

Our study is similar to those of Dondo and Cerdã [18], Moolman et al. [40] in terms of using clustering analysis to solve a capacitated vehicle routing problem. The first part of our study, i.e., the formulation of the MIP model is derived from the work done by Achutan and Caccetta [1], Christopher Mohr [38] and Toth and Vigo [53]. In the second part we develop an S-shaped routing algorithm [29] combined with a balanced K-Means algorithm of He et al. [26] to solve the vehicle routing problem. However, neither of these papers address a warehouse facility with narrow, uni-directional aisles which is what we consider for our problem.

Chapter 3

Uncapacitated Problem

In this chapter, we describe and analyse the uncapactitated case of the forklift routing problem. The chapter is divided as follows. We first define the problem and give the general description of the uncapacitated case. We then formulate the Mixed Integer Programming (MIP) model for the problem followed by our designed heuristic. We then compare the exact and heuristic methods and conclude.

3.1 Problem Statement

The warehouse management receives several orders throughout the course of a particular time period. Here we define "orders" as a set of products that are present in specific locations throughout the warehouse for the forklift to travel and pick up. The forklift can only be said to have completed an order once all the items present in the set have been picked up and brought to the depot. The problem looks at picking of an optimum route taken by the forklift in a warehouse when it receives an order to pick up. In this chapter, we ignore the capacity of a forklift. That means, a forklift can start its journey, pick up all the products ordered and return to the depot in a single tour. The goal is to minimize the order picking time, i.e., reducing travel distance thus reducing travel time and cost. However, designing the route has substantial obstacles which are described in the following subsections.

3.1.1 Warehouse Layout

The layout of the warehouse in context is a typical warehouse with equidistant shelves/aisles along with gangways/pathways and inbound/outbound area. The warehouse has a depot or a docking point where the forklift drops off the picked up items. The warehouse consists of several aisles with slots. The products are placed in these shelves. The aisles are

- Adjacent to one another
- Equidistant to one another
- Accessible through the gangways on either end of the warehouse
- Very narrow (such that U-turn is restricted) and possess no middle aisles or a gangway in between

A simple layout of the warehouse is shown in Figure 4.

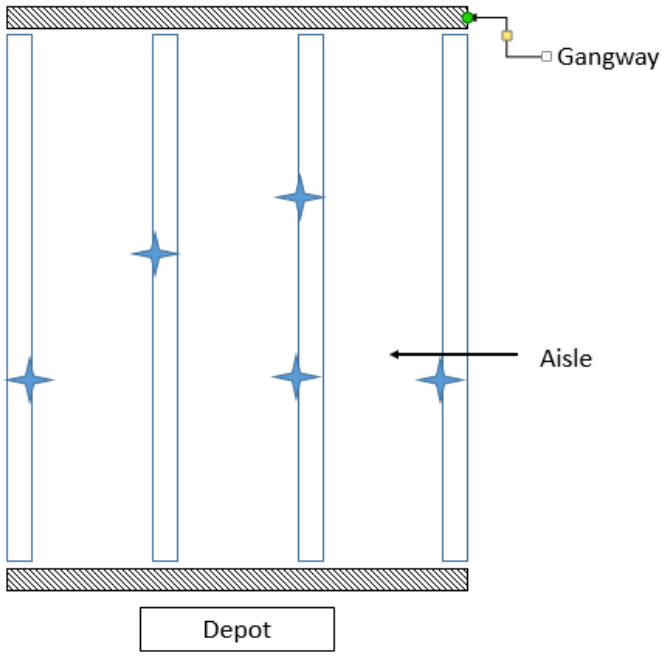


Figure 4: Simple Warehouse

Modelling the warehouse using Graph Theory

Due to the layout described in the previous section, the particular type of graph that can be used to model it is a lattice graph. Since the aisles are completely straight and none of the aisles are diagonal, there is a constant gap between each of the aisles, a lattice graph can be used to represent the layout of the warehouse. A lattice graph, also known as mesh graph or grid graph can be defined as "a graph, whose graph, embedded in some Euclidean space $\mathbb{R}^n$, forms a rectangular tiling" [54]. The entrance and the exit points of each aisle are modelled as nodes whereas the aisles themselves and the gangways can be modeled as the edges of the graph respectively. This lattice graph representation of the warehouse is shown in Figure 2. This is the layout and the graph model that will be referenced to in solving the routing problem.

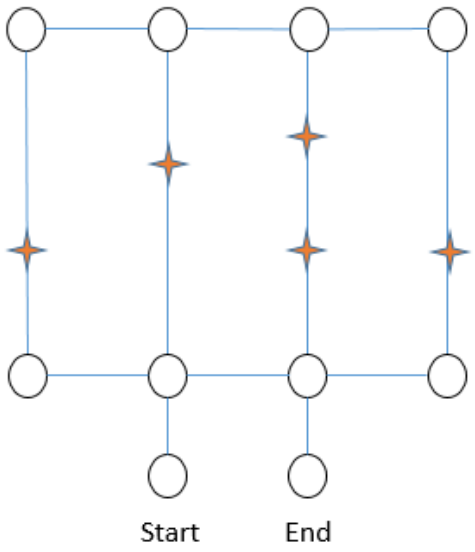


Figure 5: Lattice model of the warehouse as shown in Figure 4 (with the same set od orders)

As discussed above, the depot is modelled as two nodes of "Start" and "End" points. The graph has the geometrical property of a rectangular tiling and can hence, be labelled as a "lattice" graph. The horizontal lines are the gangways with nodes on them denoting the entry and exit points for the aisles. The marked points can be considered as the location of items of a random order.

3.1.2 Directionality

As mentioned above in section 3.1.1, one of the major difficulties is the fact that the aisles are very narrow. This narrow gap between the aisles prevents the forklift from making a U-turn for safety reasons while it is traversing the aisle. Once it has entered the aisle, it is required to traverse the entire length of the aisle and exit the aisle from the opposite end. Safety is a major concern in warehouse operations and the Environment, Health and Safety (EHS) policy of the company hinders the forklift from driving in reverse mode while it is inside the aisle. This further forces the forklift to have to exit the aisle by driving through the entire length of the aisle. The warehouse management, as a result, has forced the aisles to be unidirectional. Each aisle has a pre-assigned direction through which the forklift can enter and this direction remains unchanged for each set of ordered products.

Modeling the directionality

As discussed above, due to safety issues and unique geometry of the aisles, the forklifts can only travel in only one direction along the aisles. This is modelled by enforcing uni-directional edges when representing the aisles. The gangways however, can be traversed along both directions and are thus represented as bi-directional edges. This is illustrated in Figure 6.

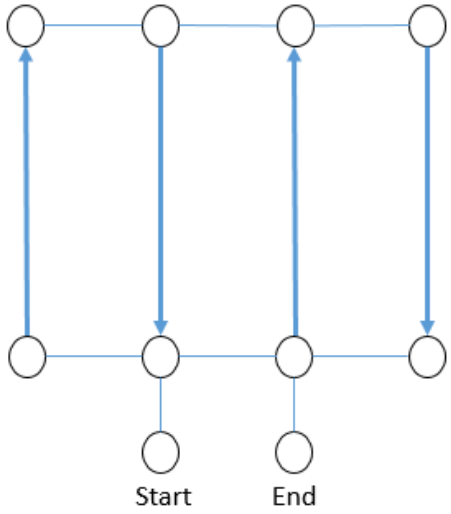


Figure 6: Lattice model of the warehouse in Figures 4 and 5 with aisle directions

3.1.3 Assumptions

In this chapter, we work under the assumption that the forklift used has an unlimited capacity and has the ability to pick up all the ordered items and return to the depot within the same tour. The following assumptions have also been made which are essential for solving the optimization problem.

1. The geographical locations of the ordered items are fixed and known.

2. The docking point is modeled as a single node. Realistically, the docking point is a platform where vehicles can arrive and arrange themselves in a line to unload the products. Since we are analyzing the case of one vehicle, it would make sense to model the docking point as a single node.

3. The forklift travels at a constant speed. This assumption also makes sense since for EHS (Environmental, Health and Safety) issues, a standard warehouse has strict restrictions on the speed that a forklift can be driven at. Moreover, since the acceleration time is negligible, the speed is considered to be uniform.

4. Only single tier aisles are considered. The objective for us is to minimize the

total time taken by the forklift to complete a tour and pick up all the products. If multiple products are at the same location in terms of aisle number and position along the aisle, but are in different vertical slots in that position,we assume them to be at the same location. We discuss the possibility of relaxing this assumption in Chapter 5 as part of future work. In this problem, we are not considered with the height of the storage shelves.

5. When being loaded on the forklift, all items can be stacked on one another and the weights of the items are not considered while stacking. The problem only looks at finding an efficient route and we are neither looking at the stackability condition of items nor are we considering any stackability matrix.We discuss the possibility of relaxing this assumption in Chapter 5 as part of future work.

Given an order consisting of a set of times that represent how far they are from the depot and assuming knowledge of the warehouse layout, the goal of the problem is to find a route that minimizes the total time needed to pick up all the products in the order after imposing the above assumptions.

3.2 MIP Formulation

The graph used to formulate the problem is the one used in the previous section (Figure 5). Let $G(V)$ be a directed graph, where $V = 1, ..., n$ is the node set that represents the items in the order-list or the points where the items are located. The node set comprises only the location of the items.

3.2.1 Graphical Representation

Before moving on to the MIP model, we describe the mapping of our directed graph into a simplified version. The simplified version is based on one of the approaches of Theys et al. [50] of simplifying a problem by computing the shortest paths between every pair of nodes.

Illustrative Example We take a scenario as shown in Figure 7 where an order contains 3 items to be picked up (all weights in kg) and the capacity of the forklift is

infinite.

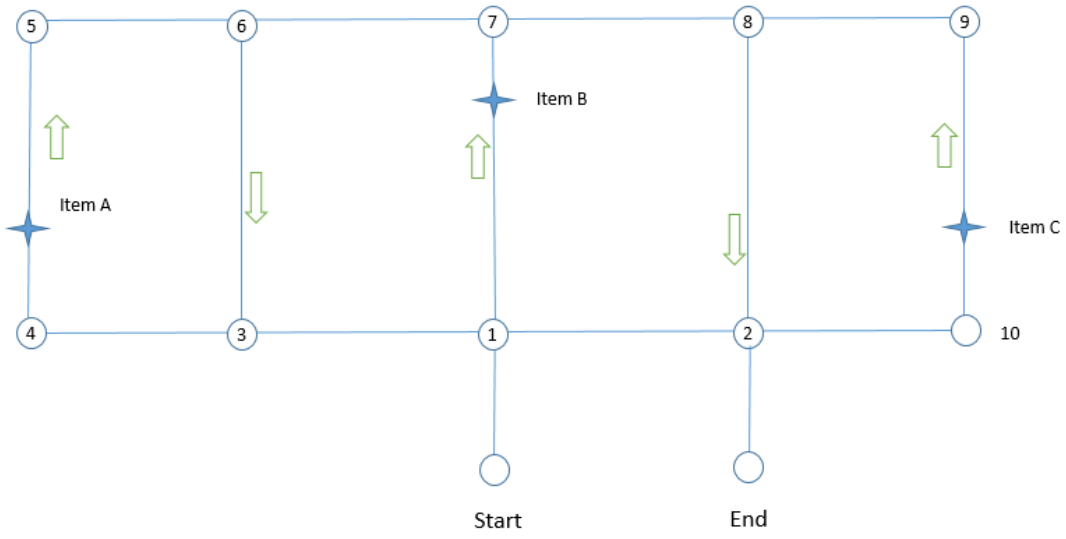


Figure 7: An example of a scenario

Figure 7 shows a 5 aisle warehouse with 3 items to be picked up. The individual weights of the items are redundant in the uncapacitated case as we are assuming that the forklift has infinite capacity and is capable of picking up all the products irrespective of their weights. The $c_{i,j}$ parameters represent the shortest distance between two points. For example, the distance value from "Start" to "Item A" is given as

$c_{Start,A}$ = (Start,1) + (1,3) + (3,4) + (4,A)

By using arcs to represent the shortest path between two item locations, the above representation can be simplified into a graph as shown in Figure 8.

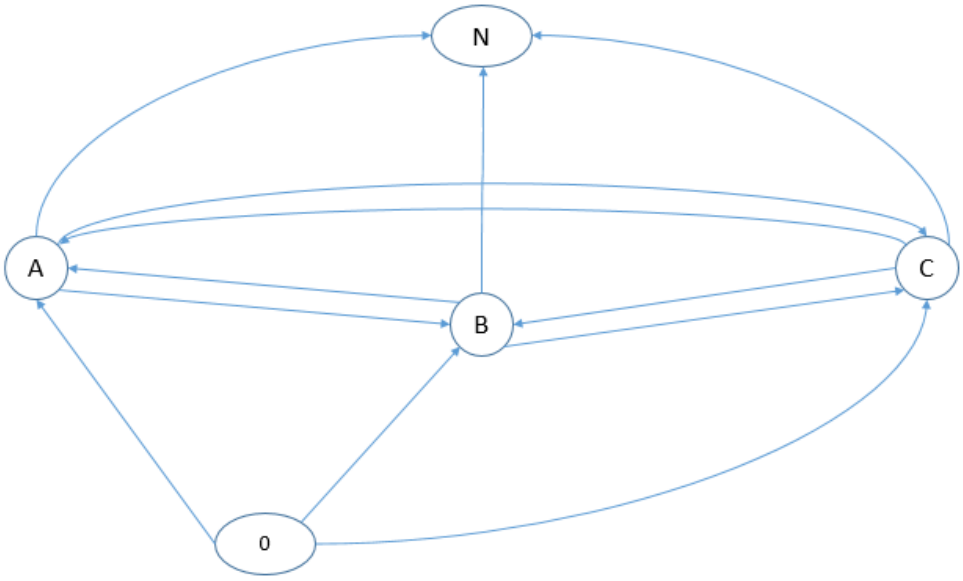


Figure 8: Simplified model of scenario

To be consistent with the model the "Start" node is replaced by 0 and the "End" node by N. The nodes A,B,C represent the item nodes. We can see that Node 0 only has leaving arcs and node N only possesses entering arcs signifying that node 0 is only for departure and node N is only for arrival. An item node, however, possesses both an entering arc and a leaving arc. The arcs themselves represent the distance between two nodes, $c_{i,j}$ parameter of the MIP model. For example the arc between node 0 and node A represents the distance between the "Start" point of the depot and the location of item A, the calculation of which is shown above. The item nodes A,B,C are interconnected to one another representing that a forklift can move from one item location to another. We walk through our MIP model while referring to Figure 8 for explanation in the latter parts of the section. This method of graphical representation only needs to be carried out once and the shortest distance of every pair can be stored in a database from which the necessary values can be extracted according to the order that arrives.

3.2.2 Objective

The objective of the problem is to find an optimal route for a given order. The first task is to define what we mean by an "optimal" route in this case. The management

is always looking to reduce the time it takes to collect all the ordered items and return to the depot, which is a complete tour. Since one of our assumptions is that the forklift travels at a uniform speed, minimizing the total distance of a route or tour is equivalent to minimizing the time taken to complete that tour. Hence, the objective of the problem is set as minimizing the total distance that the forklift has to travel in order to pick up all the order items.

3.2.3 Parameters

The parameters are stated below.

- n : The number of items ordered
- V : Node set that represents the ordered items or the points where the items are located $1, ..., n$
- $c_{i,j}$: The shortest distance between nodes i and $j \in V$
- q_i : The weight of a product i
- Q_{max} : The maximum capacity of a forklift, defined as the limit or the sum of weights that the forklift can carry in a single tour
- LB : Lower bound on the number of tours that the forklift needs to take to collect all the items, defined as

$$LB = \frac{\sum q_i}{Q_{max}}$$

The equation above is dividing the sum of weights of all the ordered products by the capacity of the forklift. For the uncapacitated case, since the maximum capacity is assumed to be very high, we practically assign LB to be the value 1.

The direction is modeled using $c_{i,j}$ parameter. It should be remembered that the distance matrix is the array of shortest distances from each point to another. The cost of traversing an aisle in the reverse direction is assigned as a very large number in order to take care of the unidirectional characteristic of the problem

3.2.4 Decision variables

We define $x_{i,j}$ as the variable to denote whether an aisle is visited or not during the course of a tour. This is modeled by assigning a binary variable to each aisle, where the value 1 would denote that the aisle is visited and the value 0 would signify that the aisle has not been visited:

$$x_{i,j} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is visited} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The variable y_i is assigned at node i to keep track of how many nodes have been visited by the forklift.This variable allows us to formulate a constraint that ensures that all ordered items are picked up.

y_i = cumulative number of nodes visited at node i.

3.2.5 Model

We base our model on the work done by Achutan and Caccetta [1], Mohr [38] and the vehicle flow model mentioned by Toth and Vigo [53]. Since we now know the parameters and decision variables associated with our problem, we can go on to state the Mixed Integer Programming model. The start point of the depot is represented by the 0 node whereas the end point is represented by N node.

$$\text{Minimize} \quad \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} \tag{2}$$

$$\text{s.t.} \quad \sum_{i \in V \setminus \{j\}} x_{i,j} + x_{0,j} = 1 \qquad \forall j \in V, \tag{3}$$

$$\sum_{j \in V \setminus \{i\}} x_{i,j} + x_{i,N} = 1 \qquad \forall i \in V, \tag{4}$$

$$\sum_{j \in V} x_{0,j} = 1 \qquad , \tag{5}$$

$$\sum_{i \in V} x_{i,N} = 1 \qquad , \tag{6}$$

$$y_j - y_i \geq (n+1) x_{i,j} - n \qquad \forall (i,j) \in V \mid i \neq j, \tag{7}$$

$$y_N \geq n + 1 \tag{8}$$

$$y_0 = 0 \tag{9}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in V, i = 0, i = N \tag{10}$$

$$y_i \geq 0 \qquad \forall i \in V \tag{11}$$

The objective function (2) minimizes the total distance traversed by the forklift to collect all the items. Constraints (3) and (4) impose flow conservation. In other words, (4) is implying that for all nodes within the set of ordered items, a forklift can either go to another order point or go to the "End" node, ie, return to the depot. Similarly, (3) implies that the edges entering any of the order nodes must either come from the "Start" node or from any of the other order nodes. Constraint (5) states that forklift can use only one arc for leaving the "Start" node since the number of tours in this uncapacitated case can only be one. Similarly constraint (6) signifies that there can be only one node entering the "End" node for the very same reason. We add constraints (7) to (11) to the model to signify that the forklift has to visit all the order nodes. (7) makes sure that the cumulative number of nodes is increasing as the forklift moves through the graph (i.e., the number of visited nodes is increasing). (8) implies that a forklift can never visit directly from start point to end point, it has to pass through all order nodes before visiting the "End" node.

In order to better understand our MIP model, we refer back to our Illustrative Example and in particular to the simplified graphical representation of Figure 8. The

objective function (2) minimizes the total distance traversed by the forklift to collect all the items. Referring to Figure 8, this is the summation of all the arcs being used to traverse in order to collect all the items A,B and C. Constraints (3) and (4) impose flow conservation. Referring to Figure 10 again, if we consider only node A we see that node A can either be reached from Node 0 or from any of the other item nodes (B and C). Similarly, any arc leaving out from node A is either directed to the node N or to any of the other item nodes (B and C). This characteristic is modeled by constraints (3) and (4). The value of LB in this case is calculated according to the formula for LB.

Since, the maximum capacity (Q_{max}) is infinity the value for LB is 1, which as we defined is the lower bound on the number of tours that the forklift needs to take to collect all the items Hence, (5), in reference to the example of Figure 8, states that in order to collect all the items (A, B and C) the forklift has to make only one tour and hence there has to be at least one arc leaving the depot (node 0). Same logic applies to the "End" node (node N) and hence is represented by (6) stating that there should be at least one arc entering node N. We now focus on the node visiting constraints. Constraint (9) states the the cumulative number of nodes visited when the forklift is at node 0 is 0. This makes sense since we are starting our tour from node 0, which is the "Start" node and there are no prior nodes where the forklift has actually visited. Constraint (8) prevents the forklift from directly visiting the "End" node from the "Start" node. The constraint states that the difference between cumulative nodes visited at node N and cumulative nodes at node 0 has to be more than or equal to n+1 number of nodes. Referring to our example, we see that this constraint translates to the difference between cumulative node at N (y_N) and cumulative node at node 0(y_0) to be at least 4. In doing so, we ensure that the forklift will not be visiting the "End" nodes until it has visited all the item nodes (which is 3) plus the node it already started from, i.e, "Start" node thus making the total 4. If we assume that the forklift is currently in node "A" and we wish to move to node "B". Constraint (7) states that while moving between the nodes that contain items the difference between cumulative node visited of the two respective nodes becomes at least one. Referring to our example, if we move from node "A" to node "B", the variable $x_{A,B}$ becomes 1. This results in the difference between cumulative node at "A" (y_A) and cumulative node at "B" (y_B) to take the value of at least 1, implying

that the forklift can immediately travel from “A” to “B”.

This is the MIP model for the uncapacitated case that we are going to solve and will refer to in this chapter.

3.3 Heuristic

The primary goal behind solving this problem is to ensure that the company can find an optimal route for its forklifts in the shortest possible time. Finding the optimal route is the objective but the solution also needs to be found in the shortest possible time. Every second the management spends in finding the optimal route is loss in profit. The MIP model is very effective in solving for small instances, but for large instances as we will see further in the Experimental Results section 3.4, the MIP model takes a long time to solve. Hence, it is paramount that we look for an alternative in a heuristic approach that is able to provide us with a near to optimal result in a very quick time. The heuristic developed in this case has several major components as part of its skeleton. It is explained according to the following stages :

- Preprocessing: described in Section 3.3.1
- Initialization: Section 3.3.2
- Iteration: Section 3.3.3
- Termination: Section 3.3.4

The complete algorithm is presented as pseudocode (Algorithm 1) at the end of Section 3.3. In Secion 3.4, we will present experimental results comparing the performance of MIP and heuristic methods.

3.3.1 Preprocessing

The first stage in solving any problem is to make the problem simplified so that it can be mathematically computable. The problem in hand is no different case. Several methods and steps are involved in our preprocessing stage which are discussed below. It is important to remember the inputs or the parameters that we have on hand once an order is received. They are, for the reader’s ease of reading, discussed below again.

- The uncapacitated case is only considered with the location of the products. Hence we can ignore the weights of the items and only consider their location.
- Narrow aisles is one of the key obstacles of the problem as previously mentioned in Chapter 1. Hence when a forklift enters an aisle it has to traverse the entire aisle. Since there is no restriction on the amount of objects that it can pick up, it only makes sense for the forklift to pick up all the ordered items in an aisle once it has entered that particular aisle. Hence, we do not need to focus on the exact location of the products, but just in which aisle the product is located. So the entire tuple of information of a product's weight and location can be simplified as we just focus on the aisle where it is located.
- All the aisles having one direction are grouped as one set and all the aisles having opposite direction are grouped as a different set. They are labeled as "Direction Up order aisles" and "Direction Down order aisles".
- Another grouped set is created for all the aisles, irrespective of containing orders or not. These are basically all the aisles divided into two sets of directions. They are labeled as "Direction Up non-order aisles" and "Direction Down non-order aisles".

Algorithm 1 PREPROCESSING

$\{Order\} \leftarrow$ all aisles that contain at least one ordered item
$\{Up\} \leftarrow$ all Up direction aisles with orders
$\{Down\} \leftarrow$ all Down direction aisles with orders
$\{Non-order\ Up\} \leftarrow$ all Up direction aisles in the layout
$\{Non-order\ Down\} \leftarrow$ all Down direction aisles in the layout

3.3.2 Initialization

A crucial part of any heuristic is deciding where to start the heuristic from. The heuristic developed here is no different. When an order arrives, the tour starts from the "Start" point of the depot. The task of the initialization phase is to decide which aisle the forklift should go to first in order to begin picking up all the products. The procedure of making this decision is discussed below.

- All the ordered products present in the "Up" direction aisles are considered. Since we are only interested with the aisles the products are present in, rather than the exact locations, we only measure the distances from the starting point of the depot to each of these selected aisles.

- The aisle that results in the largest distance from the depot is chosen. This concept relates to the fact that we want to start our tour from one end of the warehouse and then proceed on in such a way that we do not have to return to pick up an item on that side of the warehouse again.

- Choosing an aisle with "Up" direction makes sense since in order to enter the lattice network we have to enter it through an "Up" direction aisle. Hence, it is almost trivial that the best case scenario would be to actually look for those aisles where an ordered item is actually present so as to minimize the distance traveled by the forklift

- If there are no "Up" direction aisles with any ordered item present, we then opt for the farthest "Down" direction aisle containing a product. But in order to access it, we have to pass through an "Up" directional aisle first. This aisle is chosen according to the one that is closest to the "Down" direction aisle but also nearest to the depot.

Algorithm 2 INITIALIZATION

Start from "Depot"
if $\{Up\} \neq \emptyset$ **then**
  for each a in $\{Up\}$ **do**
    $dist[a] \leftarrow distance\,of\;\; aisle\;\; \text{a}\;\; from\;\; depot$
  go to a* = farthest aisle
  remove aisle from $\{Order\}$
else
  for each a in $\{Down\}$ **do**
    $dist[a] \leftarrow distance\;of\;aisle\;from\;depot$
  $auxiliary\;aisle \leftarrow aisle\;with\;max(dist)$
  for each element in $\{Non-order\;Down\}$ **do**
    $dist \leftarrow distance\;of\;aisle\;from\;auxiliary\;aisle$
  go to a* = nearest aisle
  remove aisle a* from $\{Order\}$
$decision\;aisle \leftarrow go\;to\;aisle$

3.3.3 Iteration

The initialization phase leaves us at the farthest most "Up" direction aisle or the "Up" direction aisle that is the maximum distance from the depot. This point will be defined as our "decision point" since it is at this point we will look to traverse through an opposite direction aisle. The iteration stage is summarized according to the following steps:

1. The first stage in iteration is to count the number of "Down" direction aisles on both sides of the decision point. This stage is crucial as it will help us decide where the forklift should go to next. Once they are counted, we opt for the direction which contains the least number of "Down" direction aisles. Let this be assigned as D_1. If there is only one direction to traverse, in other words, the number of "Down" aisles in any one direction is zero, we can skip to Step 2.

2. Once we have the direction to traverse next, we now choose the nearest "Down" direction aisle. After traversing that aisle, we return to the "static point", defined in the previous step again and repeat until all the "Down" direction aisles in the D_1 directions are finished.

3. Now we shift our focus to the D_2 direction. We again consider only the "Down"

direction aisles. We choose the aisle which is closest to the decision point and move there. This aisle now becomes our decision point. If there are no "Down" direction aisles, or in other words, if the "Down" direction aisle set is empty, we skip to step 5.

4. We then look at the "Up" direction aisles and choose the one that is closest to the decision point. If there are more orders left to pick up we repeat from step 3 again. If there are no "Up" direction aisles or the "Up" direction set is empty, we skip to step 6.

5. If no "Down" direction aisles are left, or the "Down" direction set is empty, we turn our attention to the non-order "Down" direction aisles and choose the one that is closest and continue with step 4.

6. If no "Up" direction aisles are present, we look at the non-order "Up" direction aisles and choose the one that is closest. We move repeat with step 3 again.

Basically, this stage is traversing all the aisles that contain the orders by adopting a strategy where we try to minimize the number of aisles traversed between two aisles containing one or more ordered items.

Algorithm 3 ITERATION

$\{D_L\} \leftarrow$ order containing down direction aisles on Left side of decision aisle
$\{D_R\} \leftarrow$ order containing down direction aisles on Right side of decision aisle
if $|D_L| \leq |D_R|$ **then**
    for each a in $\{D_L\}$ **do**
        go to a
        remove a from $\{Order\}$
        return to decision aisle
while $\{Order\} \neq \emptyset$ **do**
    if $\{Down\} \neq \emptyset$ **then**
        for each a in $\{Down\}$ **do**
            $dist[a] \leftarrow distance\ from\ current\ aisle$
        go to nearest aisle from current aisle
        remove aisle from $\{Order\}$
    else
        for each a in $\{Non-order\ Down\}$ **do**
            $dist[a] \leftarrow distance\ from\ current\ aisle$
        go to nearest aisle that is also nearest to depot
    if $\{Up\} \neq \emptyset$ **then**
        for each a in $\{Up\}$ **do**
            $dist[a] \leftarrow distance\ from\ current\ aisle$
        go to nearest aisle
        remove aisle from $\{Order\}$
    else
        for each element in $\{Non-order\ Up\}$ **do**
            $dist \leftarrow distance\ from\ current\ aisle$
        go to nearest aisle that is also nearest to depot
end while

3.3.4 Termination

The iteration stage consisted of repeating the mentioned steps. But, the algorithm has to terminate at one point. Let us remind ourselves once again that the objective is to find an optimal route in terms of least time taken to travel in collecting all the products. Hence the algorithm terminates as soon as all the products have been collected. After running the iteration stage and collecting all products, the final stage is to return to the depot.

- If the last product collected was in a "Down" direction aisle, we simply return to the depot from that specific aisle.
- If the last product collected was in a "Up" direction aisle, we choose the nearest "Down" directional aisle that is also closest to the depot. Then we return to the depot.

Algorithm 4 TERMINATION

if direction of current aisle is "Up" **then**
  for each element in $\{Non - order\ Down\}$ **do**
    $dist \leftarrow distance\ from\ current\ aisle$
  go to nearest aisle that is also nearest to depot
  return to Depot
else
  return to Depot

A complete pseudocode for the algorithm is given on the next page as Algorithm 5.

Algorithm 5 Heuristic for the Uncapacitated Forklift Setting

1: **Preprocessing**
2: **Initialization**
3: **Iteration**
4: **Termination**

	MIP model	Heuristic
Processor	2.3 GHz	2.3 GHz
RAM	4 GB	4GB
Operating System	64-bit	64-bit
Solver	CPLEX 12.7	Python 3.6

Table 1: System Requirements for Uncapacitated Heuristic

3.4 Experimental Results

Both the Mixed Integer Programming model and the Heuristic were implemented for different scenarios and their results are discussed in this section. We use the following computer system and software to run our experiments

3.4.1 Experimental Setup

The experiment is set up to test and analyse the results both from the MIP model and the heuristic. We vary our parameters according to the number of aisles and number of products. In other words, we run instances for different combinations of aisles and product numbers. The experimental setup summary is given below.

- The experiment is repeated for a set of aisles. For the purpose of this thesis, we run experiments of the uncapacitated problem for scenarios of 5, 10, 15, 20 and 30 aisles respectively.

- For each scenario of aisle, we run different instances with number of products being the variant as 5, 10, 15, 20 and 30 products. Hence each scenario is a combination of products and aisles, for example, 5 aisles with 5 products, 20 aisles with 15 products, etc. Altogether we run experiments for 25 different scenarios.

- Each scenario, that is, a combination of number of aisles and number of products has 50 instances.

- The key metrics that we are interested in finding or will be used to compare the different scenarios are mean relative error and run time.
  Relative Error Relative Error will be used to indicate the optimality gap. The

lower the value of the mean error, the closer we are to the optimal result or the best result. The relative error is calculated according to the following formula.

$$relative\,error = \frac{heuristic\,solution - best\,solution}{best\,solution} \tag{12}$$

Runtime The runtime is the total run time of our heuristic or MIP model. It will give us an idea of how fast our models are. The runtime is measured in seconds.

Since we are interested to find the optimal route but also want it to find it in the quickest time possible, it justifies to use relative error and runtime as our comparison indicators.

3.4.2 Results and Discussions

We now look at the results according to the setup described in the previous section.

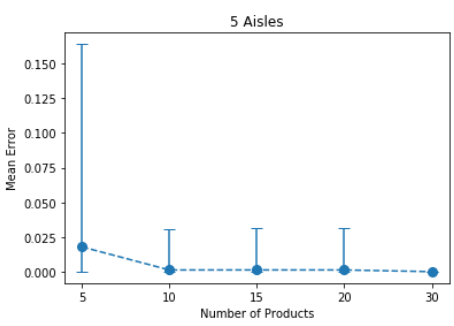
Mean Relative Error

The mean error of each instance is calculated according to the formula described before and the results are plotted below.

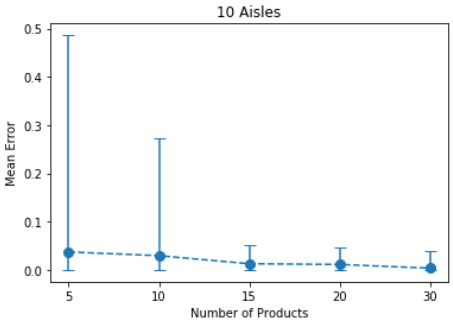
As we can see, each of the plots represent a warehouse with a fixed number of aisles. For example, plot (b) shows the scenario of a ten-aisle warehouse where we run instances for five product, ten products, etc.

- For each graph, we see that the general trend is that the mean error decreases as we increase the number of products. This implies that our heuristic is performing better as we are increasing the number of products in a fixed scenario of aisle number. This is because, since the instances are randomly generated, as we increase the product numbers they are more likely to be distributed uniformly covering a greater range of aisles than for smaller number of product. For example, consider a scenario of 5 aisles. As we increase the number of products, there is an increasing probability that the products are distributed in the same set of aisles and since for the algorithm of the uncapacitated case, we are only interested in knowing in which aisles the products are located, the algorithm ends up solving a simplified scenario and gives efficient results.

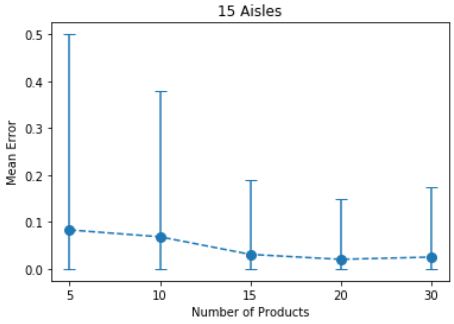
- The next point to note is the change in standard deviation. We run each scenario for 50 instances. We see from the figures that the standard deviation too decreases with increasing number of products. This too is due to the fact explained above. As the number of products keep on getting significantly larger compared to the number of aisles, we end up with instances where we have to traverse the same set of aisles due to random uniform distribution.



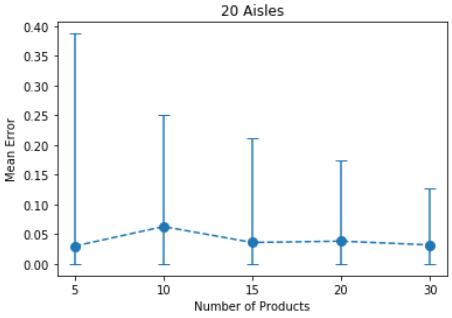
(a) Error trend for 5 aisles



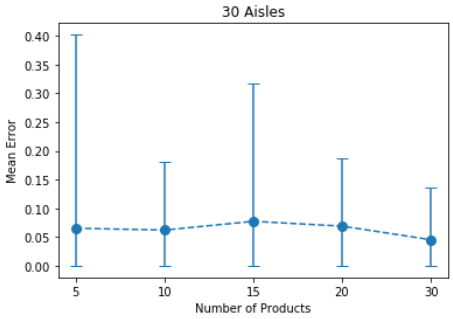
(b) Error trend for 10 aisles



(c) Error trend for 15 aisles



(d) Error trend for 20 aisles



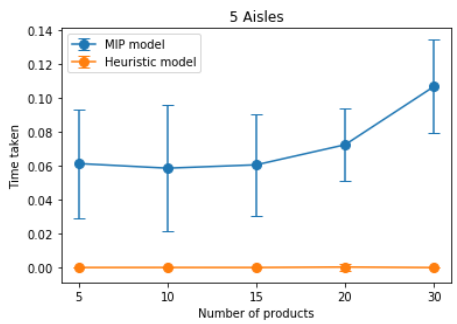
(e) Error trend for 30 aisles

Figure 9: Mean Error for different scenarios with minimum and maximum values

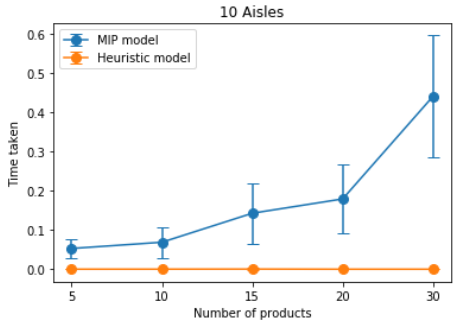
Run Time

The next parameter we compare is the runtime. This is an important factor to our model since the the current system is slow and it requires a very long time to obtain the best result. The plots for the runtime are shown in Figure 8. There are several points to be noted about the results.

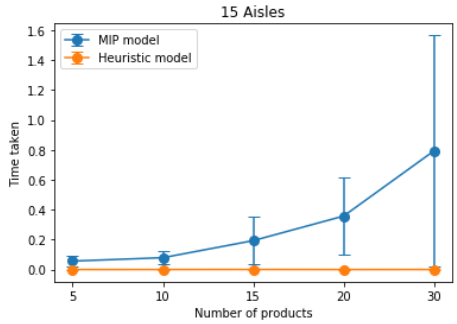
- Looking at the MIP model curve we see that as we increase the number of products in a scenario the run time increases. This makes sense since a higher number of products implies bigger network/route to cover and hence more computation by the MIP model and hence a greater runtime.
- The standard deviation of mean runtime also increases with the number of products in a scenario. This signifies that a scenario with higher number of products has more variation in its runtime than a scenario with fewer products. The reason for this is, for example, in a scenario with 30 aisles and 30 products the MIP model has a much larger number of branches to explore in a branch and cut algorithm. In some instances, it reaches optimality fairly quickly but in other instances the time can be significantly large.
- It should be noted that the heuristic algorithm is faster than the MIP model in literally all of the scenarios. Furthermore, the runtime of the heuristic stays pretty much constant compared to the MIP model, which increases significantly for larger scenarios in terms of products and aisles.
- Figure (f) perfectly depicts the relative speed of the heuristic algorithm. We can see that for smaller instance scenarios, the MIP model and heuristic algorithm are very close in terms of runtime. However, for large instances (e.g, 30 aisles and 30 products), the mean runtime for trip significantly increases where as the heuristic runtime is almost always the same.



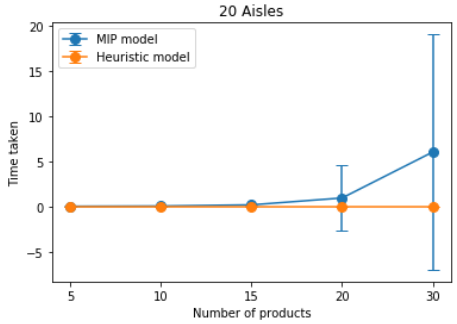
(a) Runtime (seconds) for 5 aisles



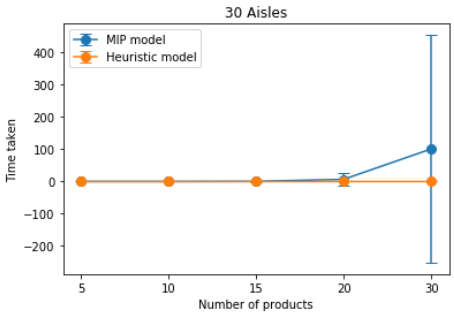
(b) Runtime (seconds) for 10 aisles



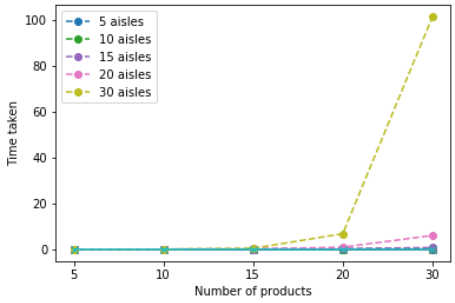
(c) Runtime (seconds) for 15 aisles



(d) Runtime (seconds) for 20 aisles



(e) Runtime (seconds) for 30 aisles



(f) Runtime (seconds)

Figure 10: Variation of runtime (seconds) for different scenarios.

3.5 Conclusion

This section has dealt with the uncapacitated case of the forklift routing problem. We developed a Mixed Integer Programming model and a Heuristic model. The heuristic was developed primarily to overcome the time complexity of scenarios with large number of aisles and products. We used mean error and runtime as our performance indicators for both MIP and heuristic. From the set of experiments performed, we saw that the heuristic developed was always faster in terms of runtime and the error only slightly changes with complex scenarios. Hence, based on the results obtained, it is safe to conclude that the heuristic is a very good alternative to the MIP model when the MIP model gets to very large scenarios and takes significant time to reach optimality.

Chapter 4

Capacitated Problem

In this chapter, we describe and analyse the capaciated case of the forklift routing problem. The chapter is divided as follows. We first define the problem and give the general description of the capacitated case. We then formulate the Mixed Integer Programming (MIP) model for the problem based on the work done by Christopher Mohr [38] and Achutan and Caccetta [1] followed by our designed heuristic based on a clustering technique. We then compare the exact and heuristic methods and finish off with conclusion and discussion.

4.1 Problem Statement

In this chapter, we extend the problem in Chapter 3 by assuming that the forklift has a *finite* capacity. Every time the capacity is reached the forklift needs to return to the depot, drop off the items and start its journey again to collect the remaining items until all the items have been collected and returned to the depot. The layout of the warehouse, the model used to define the layout and all other directionality constraints remain the same as in Section 3.1.

4.1.1 Assumptions

In this chapter, we relax the assumption that we had imposed in the previous chapter stating that the forklift had infinite capacity and had the ability to pick up all the items within the same tour. Instead, we now assume that the forklift has a specific maximum capacity beyond which it cannot sustain the load. Whenever this load is

reached, it has to return to the depot and drop off the collected items. The remaining assumptions are same as they were in the previous chapter, but they are restated below for easier reference.

1. The geographical locations of the items ordered are fixed and known.

2. The docking point is modeled as a single node. Realistically, the docking point is a platform where vehicles can arrive and arrange themselves in a line to unload the products. Since we are analyzing the case of one vehicle, it would make sense to model the docking point as a single node.

3. The forklift travels at a constant speed. This assumption also makes sense since for EHS (Environmental, Health and Safety) issues, a standard warehouse has strict restrictions on the speed that a forklift can be driven at. Moreover, since the acceleration time is negligible, the speed is considered to be uniform.

4. Only single tier aisles are considered. The objective for us is to minimize the total time taken by the forklift to pick up all the products. If multiple products are at the same location in terms of aisle number and position along the aisle, but are in different vertical slots in that position, we assume them to be at the same location. We discuss the possibility of relaxing this assumption in Chapter 5 as part of future work. In this problem, we do not consider the height of the storage shelves.

5. When being loaded onto the forklift, all items can be stacked on one another and the weights of the items are not considered while stacking. The problem only looks at finding an efficient route and we are neither looking at the stackability condition of items nor are we considering any stackability matrix. We discuss the possibility of relaxing this assumption in Chapter 5 as part of future work.

Given an order consisting of a set of items and assuming knowledge of the warehouse layout, the goal of the problem is to find a route that minimizes the total time needed to pick up all the items in the order after imposing the above assumptions.

4.2 MIP Formulation

Let $G(V)$ be a directed graph, where $V = 1, ..., n$ is the node set that represents the ordered items or, equivalently, the points where the items are located.

4.2.1 Graphical Representation

Before moving on to the MIP model, we describe the mapping of our directed graph into a simplified version.

Illustrative Example We take a scenario as shown in Figure 11 where an order contains 3 items to be picked up (all weights in kg) and the capacity of the forklift is 15 kg.

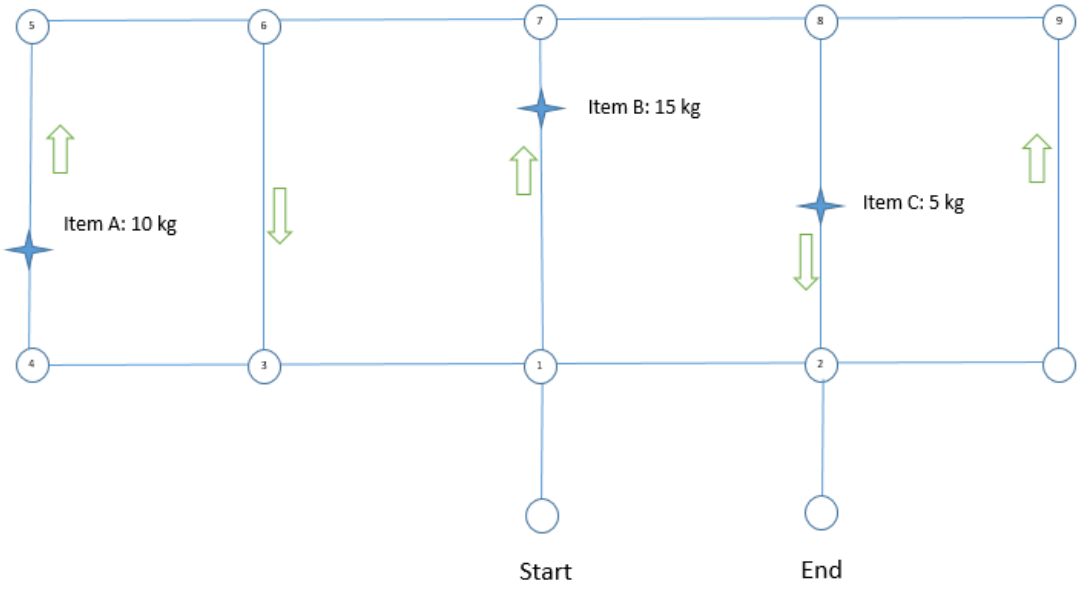


Figure 11: An example of a scenario

Figure 11 shows a 5 aisle warehouse with 3 items to be picked up. The individual weights of the items are given beside each product.The $c_{i,j}$ parameters are represented as the shortest distance between two points. For example, the distance value from "Start" to "Item A" is given as

$c_{Start,A}$ = (Start,1) + (1,3) + (3,4) + (4,A)

The above representation can be simplified into a graph as shown in Figure 12.

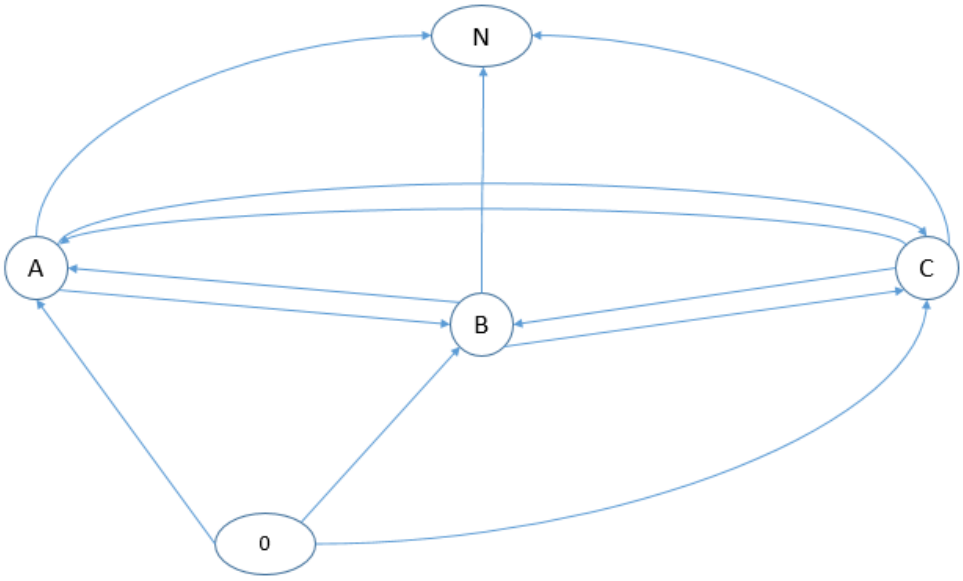


Figure 12: Simplified model of scenario

To be consistent with the model the "Start" node is replaced by 0 and the "End" node by N. The nodes A,B,C represent the item nodes. We can see that Node 0 only has leaving arcs and node N only possesses entering arcs signifying that node 0 is only for departure and node N is only for arrival. An item node, however, possesses both an entering arc and a leaving arc. The arcs themselves represent the distance between two nodes, $c_{i,j}$ parameter of the MIP model. For example the arc between node 0 and node A represents the distance between the "Start" point of the depot and the location of item A, the calculation of which is shown above. The item nodes A,B,C are interconnected to one another representing that a forklift can move from one item location to another. We walk through our MIP model while referring to Figure 12 for explanation in the latter parts of the section. As mentioned in previous chapter, this calculation for the representation only needs to be carried out once and stored in a database from which the necessary distance value can be extracted according to the order. Hence, this step has minimal effect in the processing time of the model.

4.2.2 Objective

The objective of the problem is to find an optimal route for a forklift picking up all the required items. The first task is to define what we mean by an "optimal" route in this case. As for any warehouse, the management is always looking to reduce the time it takes to pick up all the items and return to the depot, which can be defined as a complete tour. However, since the forklift has capacity, if the summation of the weights of the items exceeds the capacity, the forklift has to make more than one tour. In this case, we are looking to minimize the total time taken by the forklift to complete all the tours and pick up all of the products that are ordered. Since one of our assumptions is that the forklift travels at a uniform speed, minimizing the total distance of a tour is equivalent to minimizing the time taken to complete that tour. Hence, the objective of the problem is set as minimizing the total distance that the forklift has to take in order to pick up all the items.

4.2.3 Parameters

The parameters are stated below.

- n : The number of items ordered
- V : Node set that represents the ordered items or the points where the items are located $\{1, ..., n\}$
- $c_{i,j}$: The shortest distance between nodes i and $j \in V$
- q_i : The weight of a product i
- Q_{max} : The maximum capacity of a forklift, defined as the sum of weights that the forklift can carry in a single tour
- LB : Lower bound on the number of tours that the forklift needs to take to collect all the items, defined as

$$LB = \left\lceil \frac{\sum q_i}{Q_{max}} \right\rceil$$

The equation above is dividing the sum of weights of all the items by the capacity of the forklift.

4.2.4 Decision variables

A very important decision variable that directly affects our objective is the decision whether an aisle is visited or not during the course of a tour. This decision can be modeled by assigning a binary variable to each aisle, where the value 1 would denote that the aisle is visited and the value 0 would signify that the aisle is not visited.

$$x_{i,j} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is visited} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The variable u_i is assigned at node to keep track of how much weight has been collected by the forklift as well as how much weight remains to be collected. When the forklift arrives at a node where an order is present, it has to make a decision whether to pick up the item or not depending on the remaining capacity. This decision is made by this variable which determines the entering weight of the forklift at every node:

u_i = weight of forklift on entering node i.

4.2.5 Model

Since we now know the parameters and decision variables associated with our problem, we can go on to state the Mixed Integer Programming model below via expressions (13) to (21). The start point of the depot is represented by the node 0 and the end point is represented by node N.

$$\text{Minimize} \quad \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} \tag{14}$$

$$\text{s.t.} \quad \sum_{i \in V \setminus \{j\}} x_{i,j} + x_{0,j} = 1 \qquad \forall j \in V, \tag{15}$$

$$\sum_{j \in V \setminus \{i\}} x_{i,j} + x_{i,N} = 1 \qquad \forall i \in V, \tag{16}$$

$$\sum_{j \in V} x_{0,j} \geq LB \qquad , \tag{17}$$

$$\sum_{i \in V} x_{i,N} \geq LB \qquad , \tag{18}$$

$$\sum_{j \in V} x_{0,j} = \sum_{i \in V} x_{i,N} \qquad , \tag{19}$$

$$u_i - u_j + Q_{max} x_{i,j} \leq Q_{max} - q_i x_{i,j} \qquad 0 \leq i \neq j \leq N, \tag{20}$$

$$u_0 = 0 \tag{21}$$

$$u_i \leq Q_{max} \qquad \forall i \in V, i = 0, i = N \tag{22}$$

$$u_i \geq 0 \qquad \forall i \in V \tag{23}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall (i,j) \in V, i = 0, i = N \tag{24}$$

$$\tag{25}$$

The objective function (14) minimizes the total distance traversed by the forklift to collect all the items. Constraints (15) and (16) impose flow conservation. In other words, (16) is implying that for all nodes, a forklift can either go to another node that represents an item to be picked up or go to the "End" node, i.e., return to the depot. Similarly, (15) implies that the arcs entering any of the nodes representing an item to be picked has to be coming from another such node or the "Start" node. Constraint (17) states that there has to be at least "LB" arcs leaving the "Start" node since the lower bound on the number of tours was found to be LB. Similarly constraint (18) signifies that there can be at least LB arcs entering the "End" node for the very same reason. Constraint (19) ensures that the number of arcs leaving the "Start" point and entering the "End point" are the same, thus ensuring that all tours start and end at the same point. Constraints (20) is the subtour breaking constraint wihich also represents the capacity constraint. (21) initializes the weight of the forklift at "Start" node to be zero since there is no item present there. Constraint (22) provides

an upper bound for the weight of the forklift at a particular node to be the maximum capacity of the forklift. (23) and (24) are the non-negativity and binary constraints respectively.

In order to better understand our MIP model, we refer back to our Illustrative Example and in particular to the simplified graphical representation of Figure 12. Let us assume that the capacity of the forklift is 15kg. The objective function (14) minimizes the total distance traversed by the forklift to collect all the items. Referring to Figure 11, this is the summation of all the arcs being used to traverse in order to collect all the items A,B and C. Constraints (15) and (16) impose flow conservation. Referring to Figure 11 again, if we consider only node A we see that node A can either be reached from Node 0 or from any of the other item nodes (B and C). Similarly, any arc leaving out from node A is either directed to the node N or to any of the other item nodes (B and C). This characteristic is modeled by constraints (15) and (16). The value of LB in this case is calculated according to the formula for LB.

$LB = \left\lceil \frac{\sum q_i}{Q_{max}} \right\rceil = \lceil (10 + 15 + 5)/15 \rceil = 2$

The value for LB, in this example, comes out to be 2 which as we defined is the lower bound on the number of tours that the forklift needs to take to collect all the items Hence, (17), in reference to the example of Figure 11, states that in order to collect all the items (A, B and C) the forklift has to make at least two tours and hence there has to be at least two arcs leaving the depot (node 0). Same logic applies to the "End" node (node N) and hence is represented by (18) stating that there should be at least two arcs entering node N. We now focus on the capacity constraint (20). This constraint also acts as a subtour elimination constraint. We defined our u_i variables to be the weight of the forklift on entering node i. Hence, the weight at the very beginning of each tour, i.e., at node 0 will always be zero. This is modeled by constraint (21). In order to understand (20), let us analyse two solutions. We first analyse to see how the constraint is eliminating subtour. Suppose, there lies a solution with a subtour where we get the tours to be the following:

tour 1: "Start" - A - B - End

tour 2: B - C - B

tour 3: "Start - C - End

The second tour cannot be part of a feasible solution because it is a subtour. However, the constraint (20) ensure to remove such cases. If such a solution did exist, (20) would generate the following equations:

For tour 1:

$$u_{Start} - u_A + 15*1 \leq 15$$
$$u_A - u_B + 15*1 \leq 5$$
$$u_B - u_{End} = 15 \leq 0$$

Adding the above constraints gives

$$u_{Start} - u_{Rnd} + 45 \leq 20$$

For tour 2:

$$u_B - u_C + 15*1 \leq 0$$
$$u_C - u_B + 15*1 \leq 10$$

Ading the above constraints gives

$$30 \leq 10$$

Hence we see that there can never be any values of u_B and u_C for which the above equation will hold true. Hence, this set of tour is not possible and hence violates the constraint. Thus, such a tour is eliminated. In this way, (20) ensures that all subtours are eliminated.

Now, let us see how the equation takes care of capacity constraints. Once again, let us consider an imaginary solution with the following tours:

tour 1: Start - A - B - End
tour 2: Start - C - End

There are no subtours in this solution but as we can see tour 1 will return to the depot with a total weight of the forklift being 25 kg whereas the capacity of the forklift is 15 kg. Hence, this is clearly not a feasible solution. Let us analyse how (20) ensures to eliminate such cases.

If tour 1 exists, all the corresponding $x_{i,j}$ values will take the value 1. If that is the case, then the following sets of constraints will be generated for tour one where $x_{i,j}$=1.

$$u_{Start} \text{ - } u_A + 15 \leq 15$$
$$u_A \text{ - } u_B + 15 \leq 5$$
$$u_B \text{ - } u_{End} + 15 \leq 0$$

Adding the above wo constraints gives

$$u_{Start} \text{ - } u_E + 45 \leq 20$$

which simplifies to

$$u_{End} \geq u_{Start} + 25$$

This cannot be possible since this will be violating constraint (22) which states that the maximum value of u_i at any node is Q_{max}. Hence, it shows that such a tour is not possible. This is how the constraint removes any such tour and keeps the capacity of the forklift in check.

We now analyse the constraint (20) further to see its validity. Once again the constraint is written as

$$u_i \text{ - } u_j + Q_{max}x_{i,j} \leq Q_{max} \text{ - } q_i x_{i,j}$$

When $x_{i,j} = 1$, the equation reduces to

$$u_i \text{ - } u_j + Q_{max} \leq Q_{max} \text{ - } q_i$$
$$u_i \text{ - } u_j \leq \text{ - } q_i$$
$$u_j \geq u_i + q_i$$

The above equation means that the weight of the forklift when it enters node j from node i has to be greater than the summation of its weight at node i and the weight of the product it is supposed to pick up at node i. This makes perfect sense, and hence the constraint is valid and consitent with real-life scenario. Now let us take $x_{i,j} = 0$. We get the following reduced constraint

$$u_i \text{ - } u_j \leq Q_{max}$$
$$u_j \geq u_i \text{ - } Q_{max}$$

This constraint is also valid since we know that all u_i are non-negative variables and since u_i cannot be more than Q_{max}, the constraint is satisfied.

Since all constraints are now explained and valid, this is the MIP model that we are going to implement for our problem.

4.3 Heuristic

The primary goal behind solving this problem is to ensure that the company can find a near to optimal route for its forklifts in the shortest possible time. Every second the management spends in finding the optimal route is loss in profit due to time wasted. The MIP model can be very effective in solving for small instances, but for large instances, just as was the case for the uncapacitated problem, the MIP model can take a long time to solve. Hence, it is of paramount importance to develop a heuristic approach that would be able to provide us with a near to optimal result quickly. The heuristic developed in this case has the following major components.

- Clustering: described in Section 4.3.1
- Choosing the best cluster: Section 4.3.2
- Modifying the cluster: Section 4.3.3

The complete algorithm is presented as pseudocode at the end of Section 4.3.3. In Secion 4.4, we will present experimental results comparing the performance of MIP and heuristic methods.

4.3.1 Clustering

Due to the constraint of forklift capacity, it may not be possible to pick up all the products in a single tour.Thus, deciding which products to pick up on single tour is a major challenging feature of the capacitated problem. One way to do this would be by exhaustive search, ie checking the combination of all the n products in the order. A major drawback of this method is that as n increases, the number of combinations would increase significantly, and thus increasing the processing time. To overcome this drawback, we use a clustering algorithm based on the location of the products in the warehouse. The clustering algorithm applied is a variant of the popular K-means algorithm [47]. The following sections describe the features, their design and the steps for the K-means clustering method.

Why K-Means Clustering?

We assume that in order for a tour to be optimal the best case is that items are near to one another, but also their total weight does not exceed the capacity of the

forklift. It only makes sense for the forklift to look to fill up its capacity as much as possible with products that are geographically located near to one another in order for it to travel the least distance. In order to make clusters of such products, we decided to implement a clustering algorithm. In machine learning literature, a problem can fall under three broad categories- supervised learning, unsupervised learning and reinforcement learning [37]. The description of the three categories are too broad to discuss and beyond the scope of this dissertation. However, the problem that we are concerned with falls under the category of unsupervised learning. Broadly unsupervised learning is "a branch of machine learning that learns from test data that has not been labeled, classified or categorized [25]. For our problem, we do not have any prior knowledge as to which product should be collected in which tour. The only knowledge we do possess is a lower bound on the number of tours necessary to collect all the products.

There are many techniques in unsupervised learning literature for clustering. We choose the K-Means algorithm for the following reasons:

- The K-Means algorithm is easy and simple to implement. We are only using the K-Means as an initial starting point for our cluster of products. Most other sophisticated algorithms are much harder to implement efficiently and have many more parameters to set.
- The time complexity of K-Means makes it a great method to use in this case. One of the prime criterias of our heuristic is time and the K-Means is pretty fast compared to other clustering techniques. The time complexity of K-Means is $O(knm)$ where k is the number of centers, n is the datapoints and m is the dimensionality of data [47]. All these will be further discussed in this chapter.
- It uses random re-starts to get better local optimum.Hence, there is lower probability to get stuck in a local minima/maxima.

Feature Design

For any unsupervised learning algorithm, it is important to design a good feature set. For our problem, the features are chosen to be the shortest distance between each pair of items. The clusters formed by our heuristic will be the starting point for our routing heuristic. An ideal cluster of products should be in minimum geographical proximity to one another as well as the sum of all the products in the cluster is to be within the maximum capacity of the forklift. In order to have nearby items in the same cluster, we find the distance between every pair of products in the order. If we have n products in the order, the shortest distance between every pair of items is calculated and recorded as a matrix below where $d_{i,j}$ is the shortest distance between product i and product j.

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \dots & d_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \dots & d_{nn} \end{bmatrix}$$

At this stage, we have considered only the distances between the products. This form of the matrix can result in two very heavy products to be in the same cluster. This situation is undesirable if the summation of the two weights exceed the maximum capacity of the forklift. Hence, in order to avoid such situations the matrix is modified. If the summation of the weights of the two products exceeds the maximum capacity of the forklift, the distance between the two products is multiplied by a large number, M. This results in the two products to be placed far apart in the Euclidean space and when K-Means is run, these items will likely be in separate clusters which is desirable. For instance, if product 1 and product 2 have weights whose summation exceed the maximum capacity of the forklift, the matrix would look like the following

$$\begin{bmatrix} d_{11} & d_{12} * M & d_{13} & \dots & d_{1n} \\ d_{21} * M & d_{22} & d_{23} & \dots & d_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \dots & d_{nn} \end{bmatrix}$$

This is the feature matrix that is used to run the K-Means algorithm.

Algorithm 6 FEATURE DESIGN

```
n ← number of products ordered
products ordered ← {p_1, p_2, ..., p_n}
for each item i in products ordered do
    for each item j in products ordered do
        distance ← shortest distance(i, j)
        if distance > Q_max then
            feature value ← distance × M
        else
            feature value ← distance
```

K-Means Algorithm

The K-Means algorithm uses iterative refinement to produce a final result. The algorithm inputs are the number of clusters K and the data set, which in this case is the feature matrix above. The algorithm starts with initial estimates for the K centroids, which can either be randomly generated or randomly selected from the data set. The algorithm then iterates over two steps [47]:

1. Data assignment step: Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance. More formally, if c_i is the collection of centroids in set C_i then each data point p is assigned to a cluster based on

$$\arg\max_{c_i \epsilon C} dist(c_i, p)^2$$

   where $dist()$ is the standard (L_2) Euclidean distance. Let the set of data point assignments for each i^{th} cluster centroid be S_i.

2. Centroid update step: In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

$$c_i = \frac{1}{|S_i|} \sum_{p_i \epsilon S_i} p_i$$

The algorithm iterates between steps one and two until a stopping criteria is met (i.e, no data points change clusters, the sum of the distance is minimized, or some maximum number of iterations is reached).

K-Means Pseudo code

The K-Means clustering algorithm [47] used is summarized in the form of pseudo code below: χ : Set of points p $\{p_1, p_2, \ldots, p_k\}$
Input: Data points D, number of clusters k
Step 1: Arbitrarily choose an initial k centers $\zeta = \{c_1, c_2, \ldots, c_k\}$
Step 2: For each $i \epsilon \{1, \ldots, k\}$, set the cluster C_i to be the set of points in χ that are closer to c_i than they are to c_j for all $j \neq i$
Step 3: For each $i \epsilon \{1, \ldots, k\}$, set c_i to be the center of mass of all points in C_i: $c_i = \frac{1}{|C_i|} \sum_{p \epsilon C_i} p$
Step 4: Repeat Steps 2 and 3 until ζ no longer changes.
Output: Data points with cluster members.

It is standard practice to choose the initial centers uniformly at random from χ [47]. For Step 2, ties may be broken arbitrarily.

Determining the value of K

The algorithm described above finds the clusters and data set labels for a particular pre-chosen K. The parameter K in K-Means clustering denotes the number of clusters that the data will be divided into. It is a parameter that needs to be set before the algorithm is started. In general, there is no method to determine the exact value of K but in the case of our problem, K can be the number of tours that need to be made by the forklift to collect all the items ordered.

Since the capacity of forklift and the total weight of all the ordered items are known, the number of tours can be easily found. We are concerned with finding the combination of products that need to be collected in each tour. In other words, each tour consists of a cluster of products that are to be collected. Hence we approximate the number of clusters, K.

Optimizing centroid initialization

As mentioned in the pseudo code, the first step in K-Means is the random initialization of K centroids. This stage is crucial for the clustering in our application, since product being assigned to one cluster or the other could greatly affect the final heuristic solution. Although K-Means offers no accuracy guarantees, its simplicity and speed are very appealing in practice. However, optimizing the initialization of k centroids would give us a better chance to end up with good clusters. But, it is also important that this optimization phase be quick. Therefore, we use a variant of K-Means known as K-Means++, an O(log k)-competitive algorithm which augments K-means with a randomized seeding technique [3]. It is known that this augmentation improves both the speed and the accuracy of K-Means, often quite dramatically. K-Means++ is a specific way of choosing centers for the K-Means algorithm. In particular, let $S(x)$ denote the shortest distance from a data point to the closest cluster center we have already chosen. Then, K-Means++ can be summarized as the following pseudo code.

Step 1: Initialization of k centroids

- 1a. Take one center c_1, chosen uniformly at random from χ
- 1b. Take a new center c_i, choosing $x \, \epsilon \, \chi$ with probability

$$\frac{S(x)^2}{\sum_{x \epsilon \chi} S(x)^2}$$

- 1c. Repeat Step 1b. until we have taken k centers altogether

Step 2 to Step 4: Proceed as with the standard K-Means algorithm

4.3.2 Choosing the best cluster

In the clustering phase, we apply the K-Means algorithm to separate the products into clusters based on their geographical locations and weight. Once this phase is done, we are left with k clusters of products. However, since each cluster represents a tour, the forklift can only visit one cluster at a time. Hence, we need to select one cluster for the forklift to visit. This selection is done in this step.

The first task is to define what we actually mean by a "good" cluster. Ideally, we want to collect as many product as possible into the forklift, respecting its capacity, but we also want the products to be close to one another since our final objective is to obtain the minimum distance needed to travel. Hence we define a ratio called the compactness ratio of a cluster as the ratio between distance traveled to collect all the products in a cluster and the total weight of all the products in that cluster.

$$compactness\ ratio_{cluster_k} = \frac{total\ distance\ travelled\ in\ cluster\ k}{total\ weight\ of\ cluster\ k}$$

The compactness ratio is a measure of the distance required to collect per unit weight of product. The lower this ratio, the better the cluster since it signifies that products whose collective weight is large are geographically located close to one another and can be collected in the same tour. So this step of the heuristic involves finding the compactness ratio for each cluster and then selecting the "best" cluster which has the lowest compactness ratio.

Estimating the total distance

As described above, in order to find out the compactness ratio of a cluster we need the total distance traveled by the forklift to collect all the products in that cluster. This distance is a rough estimate since we are only interested with the comparative value rather than the exact value. At this stage, we are only interested in identifying the "best" cluster more than anything else. We assume that we will reach the same conclusion in the compactness ration whether we use the exact distance or the estimated distance. Hence, for ease of calculation we opt to use the estimated distance.

- The first step involves finding the number of aisles in the clusters containing products with "Up" and "Down" directions respectively.
- The total weight of the products in the cluster is then found along with the number of tours required to collect all the products in this cluster. This is given the name of "capacity tour" since it gives an upper bound of the number of tours needed to fill the forklift to its maximum capacity.

$$capacity\ tour = \left\lceil \frac{\sum w_i}{Q_{max}} \right\rceil$$

q_i = weight of product i

- Every time the forklift traverses an "Up" aisle, it has to traverse a corresponding "Down" aisle. We call this combination of "Up" and "Down" aisles a cycle. If the difference between the number of "Up" aisles and "Down" aisles is greater than two, "cycle" is assigned as the minimum count of "Up" and "Down" aisles. Otherwise, it is assigned as the maximum value between the two.

- The difference between the "capacity tours" and "cycles" is then evaluated.
  $|U|$: number of "Up" direction aisles in the cluster
  $|D|$: number of "Down" direction aisles in the cluster

1: $|U| \leftarrow$ number of "Up" direction aisles
2: $|D| \leftarrow$ number of "Down" direction aisles
3: **if** $|U - D| > 2$ **then**
4: $\quad cycles \leftarrow min\{U,\ D\}$
5: **else**
6: $\quad cycles \leftarrow max\{U,\ D\}$

The algorithm for choosing the best cluster is

Parameters

L = length of an aisle

h = distance between two aisles

Algorithm 7 Summary of Choosing Best Cluster

procedure CHOOSE BEST CLUSTER
  cluster weight ← sum of all products in cluster ($\sum w_i$)
  up count ← $\left|\{U\}\right|$
  down count ← $\left|\{D\}\right|$
  capacity tour ← $\left\lceil \frac{\sum w_i}{Q_{max}} \right\rceil$
  if $|up\ count - down\ count| > 2$ **then**
    $cycles \leftarrow min\{up\ count,\ down\ count\}$
  else
    $cycles \leftarrow max\{up\ count,\ down\ count\}$
  $difference \leftarrow$ *capacity tour* - *cycles*
  if $difference > 0$ **then**
    distance inside cluster ← $(2L+2h)\times difference + 2L\times cycles + (\mathbf{max}(aisles\ present) - \mathbf{min}(aisles\ present))\times h\times 2$
  else
    distance inside cluster ← $2L\times cycles + (\mathbf{max}(aisles\ present) - \mathbf{min}(aisles\ present))\times h\times 2$
  total distance ← *distance inside cluster* + (*depot to cluster* + *cluster to depot*)×*capacity tour*
  ratio ← $\frac{total\ distance}{cluster\ weight}$
  best cluster ← cluster with $min(ratio)$
  for *each item* **in** *best cluster* **do** *item weight* ← weight of *item*
  cluster weight ← $\sum$ (*item weights*)

At the end of this stage, we are left with one cluster that, according to our definition of compactness ratio, is the best cluster to visit. However, it is possible for the cluster to have a total weight that exceeds the maximum capacity of the forklift. Hence, such a cluster might need to be modified in order for the total weight to be within the capacity of the forklift.

4.3.3 Modifying the cluster

This stage will be carried out only if the total weight of a cluster found in the previous step of the algorithm exceeds the capacity of the forklift. The goal in this case is to bring down the total weight below the capacity so that the forklift can collect all the products within the cluster.

The modification stage involves removing products from the cluster until the aggregate weight of the cluster falls below the maximum capacity. But, there are certain

constraints. While removing the products, we have to be careful that we do not remove too many products so as to make the cluster contain too few products, which otherwise would contradict the original purpose of forming the cluster. Furthmore, we need to ensure that subsequent clusters that are formed would also be "good". The process of "product drop" selection is depicted below.

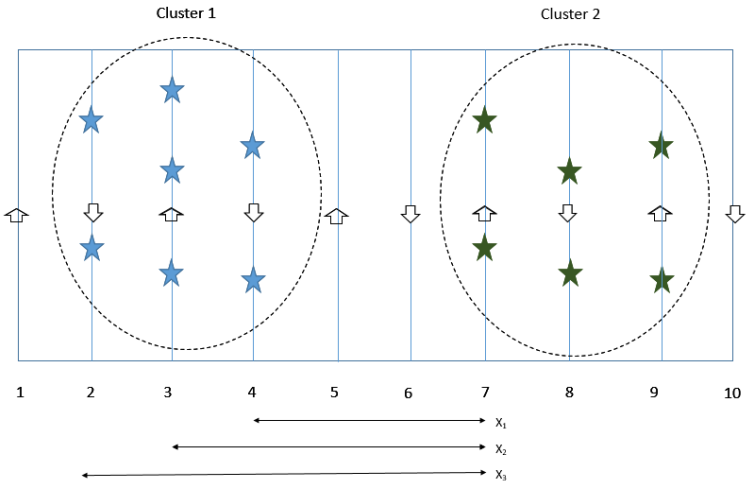


Figure 13: Cluster modification

Figure 13 shows an instance where we have two clusters. Assume that cluster 1 is the cluster that has been found to be the "best cluster" after the compactness ratio comparison. But, having calculated the aggregate weight of the products inside the cluster, it is found that cluster 1 exceeds the maximum capacity of the forklift. In such a case, we need to remove products from the cluster in order to make its aggregate weight feasible.

It is important to remember that our final objective is to minimize the total distance traveled by the forklift to collect all the products in the order set. To achieve this objective, it is important not only to have a good cluster at the current iteration, but also to ensure that the clustering phase in the next iteration has a "well structured" scenario. For example, according to figure 9, if we are dropping products form cluster 1 it is best to remove the products from the aisle that is closest to the adjacent cluster. In this case, cluster 2 consists of aisles 7, 8, 9 and cluster 1 consists of aisles 2, 3, 4 that have distances of x_1, x_2 and x_3 respectively where we can see that x_1 is the smallest of the three. Hence, we will prioritize the removal of products from the aisles according to this distance. On the other hand, if we had removed a product from the aisle 2 having the largest distance x_3, the products left for clustering in the

next iteration would be widespread and we would end up having clusters where we had to travel greater distances. Hence by removing product(s) from the aisle 4, we ensure that product(s) left for next clustering phase are close to one another to avoid having to travel large distances.

A pseudo code of this step is given below.

Algorithm 8 Summary of Modifying a Cluster

```
1: procedure MODIFY CLUSTER
2:     aisles here ← aisles in cluster containing products
3:     other aisles ← aisles outside cluster containing products
4:     extreme aisles ← two corner aisles in cluster
5:     for each element in extreme aisles do
6:         for each aisle in other asiles do
7:             distance ← distance between element and aisle
8:     choose min(distances)
9:     chosen aisle ← extreme aisle with min(distances)
10:    n ← number of items in chosen aisle
11:    for each item in choses aisle do
12:        item weight ← weight of item
13:    S ← set of item weights
14:    combinations ← (S choose n)
15:    for each combination in combinations do
16:        combination weight ← ∑(weights in combination)
17:        cluster weight ← Q_max − combination weights
18:    for each cluster weight in cluster weights do
19:        if cluster weight ≤ Q then
20:            consider cluster weight
21:    choose max(cluster weights)
22:    remove item from extreme asile not in combination
23:    return reduced cluster
```

Once a "good" cluster has been obtained based on the criteria discussed above we can then implement the "Uncapacitated problem" heuristic developed in chapter 3 on the cluster. The idea is to solve each cluster as an uncapacitated problem. This is because as long as the sum of the weights of the products in the cluster is less than or equal to the maximum capacity of the forklift, it is as if we are solving an uncapacitated instance of the problem.

Algorithm 9 Call Uncapacitated Heuristic

tour distance ← **uncapacitated heuristic**(*best cluster*)
for each item in *best cluster* **do**
    remove item **from** *products ordered*
total distance ← $\sum$(*tour distances*)

The above steps are repeated until all the products have been collected. A complete summary of the algorithm is described in the form of the following pseudo code.

Algorithm 10 Complete Heuristic for Capacitated problem

Feature Design
while *products ordered* $\neq \emptyset$ **do**
    Perform K-Means Clustering
    clusters ← **K-Means++**(*feature value*)
    Choose best cluster
    for *each item* **in** *best cluster* **do**
        item weight ← weight of *item*
    cluster weight ← $\sum$ (*item weights*)
    if *cluster weight* $> Q_{max}$ **then**
        Modify the cluster
    else
        pass
    Call Uncapacitated Heuristic
total distance ← $\sum$(*tour distances*)

4.4 Experimental Results

Similar to chapter 3, both the Mixed Integer Programming model and the Heuristic were evaluated for different scenarios. We use the computer system and software as shown in Table 1 to run our experiments

	MIP model	Heuristic
Processor	2.3 GHz	2.3 GHz
RAM	4 GB	4GB
Operating System	64-bit	64-bit
Solver	CPLEX 12.7	Python 3.6
Library	N/A	Sci-kit Learn

Table 2: System Requirements for Capacitated Heuristic

4.4.1 Experimental Setup

The experimental setup summary is given below.

- The experiment is repeated for a set of aisles. For the purpose of this thesis, we run experiments for scenarios of 10, 20, 30 and 50 aisles respectively.
- For each scenario for the number of aisles, we consider 5, 10, 20, 30 and 50 products. Hence each scenario is a combination of products and aisles, for example, 5 aisles with 5 products, 20 aisles with 30 products, etc. Altogether we run experiments for 20 different scenarios.
- Each scenario, that is, a combination of number of aisles and number of products has 50 instances. These instances are generated by generating items to be at a location, i.e., aisle number and location of the item in the particular aisle according to a certain distribution (described below).

Performance Metrics

The key factors that we are interested in finding or will be used to compare the different scenarios are mean relative error and run time.

Mean Relative Error Mean Error will be used to indicate the optimality gap. The lower the value of the mean relative error, the closer we are to the optimal result or the best result. The mean relative error is calculated according to the following formula.

$$mean\,relative\,error = \frac{best\,solution - heuristic\,solution}{best\,solution} \tag{26}$$

Runtime The runtime is the total run time of our heuristic or MIP model. It will give us a notion of how fast our models are compared to the other. The runtime is

measured in seconds.

Distribution of distances between products

The generated instances in this section contain products that are assumed to be distributed according to a particular distribution. We run instances where the products are distributed according to two different distributions.

- **Uniform Distribution:** Here, the products are distributed according to a uniform distribution where the probability of a product being in an aisle is same for all the aisles. The uniform distribution generates instances where products are placed similar to the as shown in Figure 14.

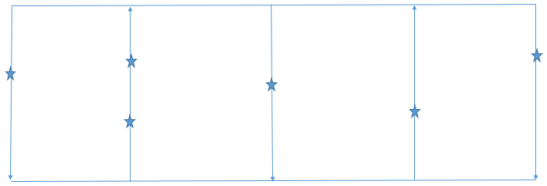


Figure 14: Uniform Distribution

- **Skewed Distribution** Here, the products are distributed so that majority of the products are skewed towards one end of the warehouse. The skewed distribution generates instances where products are placed similar to as shown in Figure 15.

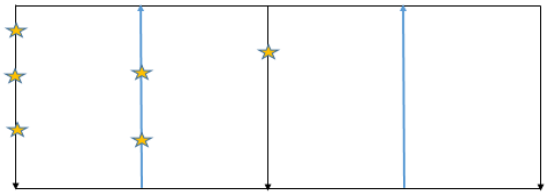


Figure 15: Skewed Distribution

This distribution is generated by assigning greater weights to the extreme aisle as shown in 15. This implies that the instance will have a greater probability of having an order on the end aisles compared to the uniform distribution where the probability of the order being in any one of the aisles was equal.

4.4.2 Results and Discussions

We now look at the results according to the setup described in the previous section.

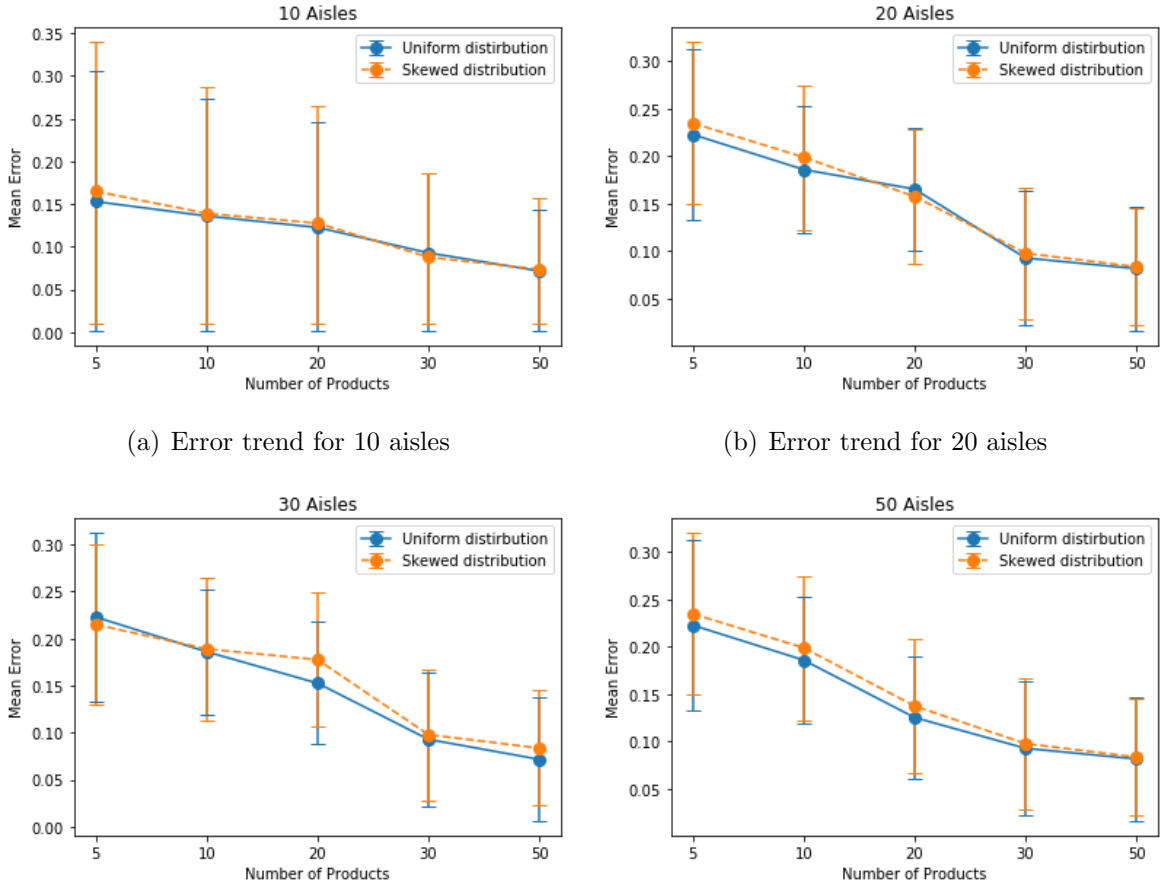
Mean Relative Error

Figure 16 shows the mean relative error for a warehouse with a certain number of aisles as the number of products increases.As we can see, plot 16 (a) to 16 (d) represent a warehouse with a fixed number of aisles. For example, plot (b) shows the scenario of a twenty-aisle warehouse where we run instances for five product, ten products, etc.

- For each graph, we see that the general trend is the decrease in mean error with the increase in number of products for a given number of aisles. This is because as we increase the number of products for the same maximum capacity of the forklift, we end up with clusters that are equal in total weight and less clusters containing only one or two products. This allows the forklift to fill its capacity as much as it can when it is choosing a cluster and thus implies that there are significantly less number of products in the latter clustering stages. Moreover, the more the number of products the more the data available for the K-Means algorithm to process, which means that the algorithm has a better chance of finding a pattern and generating better clusters which is the general requirement for a unsupervised learning algorithm to run better.

- The next point to note is the change in standard deviation. We run each scenario for 50 instances. We see from the figures that the standard deviation does not differ too much along with the increase in number of products.

- We get a similar pattern graph for skewed distribution as well. This suggests that the model performs consistently for both the uniform and skewed distribution. Even if all the products are skewed to one side of the warehouse, we can see that the model is behaves in the same way if the products were distributed uniformly.



(a) Error trend for 10 aisles

(b) Error trend for 20 aisles

(c) Error trend for 30 aisles

(d) Error trend for 50 aisles

Figure 16: Mean Error for different scenarios with minimum and maximum values

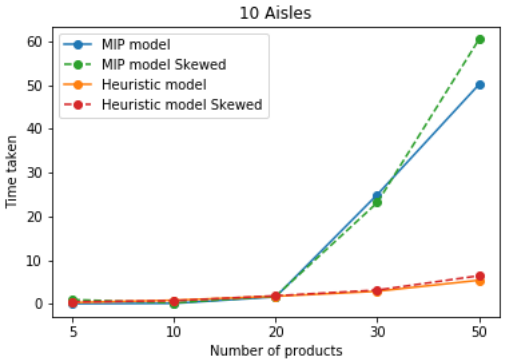
Run Time

The next parameter we compare is the runtime. Figure 17 shows the runtime plots. There are several points to note about the results.

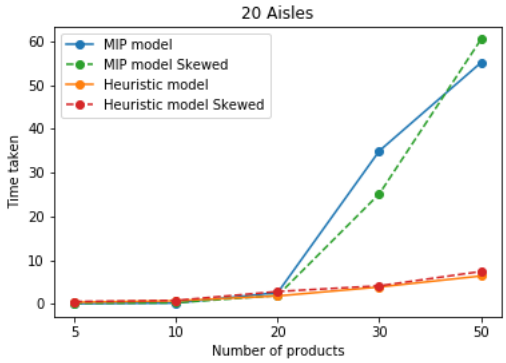
- Looking at the MIP model line we see that as we increase the number of products in a scenario the run time increases rapidly. This makes sense since a higher number of products implies more computation by the MIP model and hence a greater runtime.

- It should be noted that the heuristic algorithm and the MIP model have almost the same runtime for small instances. However, as we keep on increasing the number of products for a particular set of aisles, the MIP model takes exponentially longer time to reach optimal. The rate of increase in runtime for the heuristic, however, is very low. Hence even for instances where there are large number of products, in terms of runtime, the heuristic is much faster than the MIP model.

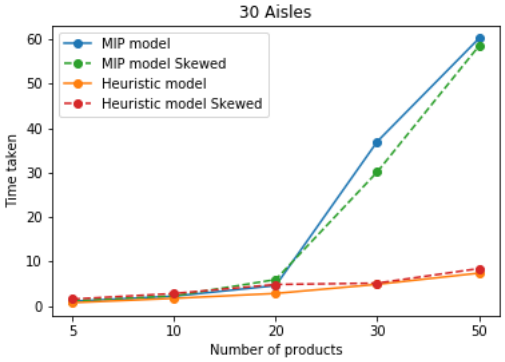
- We can see that for smaller instance scenarios, the MIP model and heuristic algorithm do not vary by a great extent in terms of runtime. However, the mean runtime for the MIP significantly increases when there are more than 30 products, while the heuristic runtime increases only slightly.



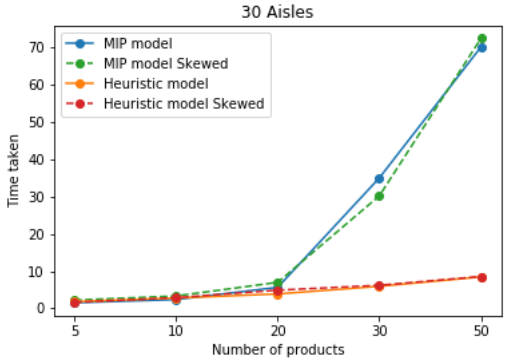
(a) Runtime (seconds) for 10 aisles



(b) Runtime (seconds) for 20 aisles



(c) Runtime (seconds) for 30 aisles



(d) Runtime (seconds) for 50 aisles

Figure 17: Variation of runtime (seconds) for different scenarios.

4.5 Conclusion

This chapter has dealt with the capacitated case of the forklift routing problem. We developed a Mixed Integer Programming model and a Heuristic model. The heuristic was developed primarily to overcome the time complexity of scenarios with large number of aisles and products. We used mean error and runtime as our performance indicators for both MIP and heuristic. From the set of experiments performed, we saw that for large instance sizes, the MIP model took a long time to reach optimality. However, the heuristic is able to obtain a good solution in comparative much faster times. Hence, based on the results obtained, we conclude that the heuristic is a good alternative to the MIP model for problems of size greater than 30 products.

Chapter 5

Conclusions and Future Work

5.1 Concluding Remarks

In this thesis we have proposed a formulation for the capacitated narrow aisle-order picking problem that is a special case of the well-known Capacitated vehicle Routing Problem (CVRP). We have been able to use classical CVRP MIP model to solve this problem arising in warehousing operations of our industry partner. However, the MIP model tends to take large time to solve large instance. Hence, we developed a heuristic approach using clustering of products and S-shaped exact heuristic to solve large instances to near-optimality in short time. We have successfully shown that the developed heuristic is a very good alternative to the MIP model both in capacitated and uncapacitated case when the MIP model becomes large and takes significant time to reach optimality.

5.2 Future research directions

To extend the current direction of this thesis, we separately suggest possible improvements for chapters. In chapters 3 and 4, further work includes modifying the clustering algorithm so that the final clusters are optimal in terms of product allocation to tours. This would ensure that all products that are possible to be collected in the next tour have been clustered in such a way that the final total distance traveled is minimum. This modification can be done in the form of designing a reward system to the clusters when removing items for clusters that are over-capacity. This kind of

reward system cn be traced to the realms of Reinforcement Learning which could be a good alternative algorithm for the clustering approach and a good starting direction in terms of future work.

In this thesis, we have implemented the K-Means algorithm for the clustering stage. However, there are many other unsupervised learning clustering algorithms that can be implemented for the clustering phase along with other exact heuristics other than S-shaped heuristics. Using other clustering techniques and exact heuristics and comparing the results with the current ones is recommended as a future research direction.

Chapters 3 and 4 of this thesis solve the models without considering the height of the storage shelves. Modifying the model as an attempt to relax this assumption can be a direction for future research along with taking into account a stackability matrix which will be used to track the product that can be stacked on top of each other.

Bibliography

[1] NR Achuthan and L Caccetta. Integer linear programming formulation for a vehicle routing problem. *European Journal of Operational Research*, 52(1):86–89, 1991.

[2] SP Anbuudayasankar and K Mohandas. Mixed-integer linear programming for vehicle routing problem with simultaneous delivery and pick-up with maximum route-length. *International Journal of Applied Management and Technology*, 6(1):2, 2008.

[3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] Barrie M Baker and MA Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

[5] Enrique Benavent and David Soler. The directed rural postman problem with turn penalties. *Transportation Science*, 33(4):408–418, 1999.

[6] Jean Berger and Mohamed Barkaoui. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & operations research*, 31(12):2037–2053, 2004.

[7] Z Boyzer, A Alkan, and A Fığlalı. Cluster-first, thenroute based heuristic algorithm for the solution of capacitated vehicle routing problem. *Int. J. Inf. Technol*, 7(2):29–37, 2014.

[8] José Brandao and Alan Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European journal of operational research*, 100(1):180–191, 1997.

[9] Mackenna Callum. A new approach for solution of vehicle routing problem with hard time window: an application in a supermarket chain. *Sādhanā*, 42(12):2067–2080, 2017.

[10] Hadrien Cambazard and Nicolas Catusse. Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane. *arXiv preprint arXiv:1512.06649*, 2015.

[11] Thomas Chabot, Leandro C Coelho, Jacques Renaud, and Jean-François Côté. Mathematical model, heuristics and exact method for order picking in 3d narrow aisles. *Journal of the Operational Research Society*, 69(8):1242–1253, 2018.

[12] Marco Colombi and Renata Mansini. New results for the directed profitable rural postman problem. *European Journal of Operational Research*, 238(3):760–773, 2014.

[13] Teodor Gabriel Crainic, Simona Mancini, Guido Perboli, and Roberto Tadei. *Clustering-based heuristics for the two-echelon vehicle routing problem*, volume 46. CIRRELT Montreal, 2008.

[14] Fabrizio Dallari, Gino Marchet, and Marco Melacini. Design of order picking system. *The international journal of advanced manufacturing technology*, 42(1-2):1–12, 2009.

[15] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[16] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen. Design and control of warehouse order picking: A literature review. *European journal of operational research*, 182(2):481–501, 2007.

[17] René De Koster, Kees Jan Roodbergen, and Ronald Van Voorden. Reduction of walking time in the distribution center of de bijenkorf. In *New trends in distribution logistics*, pages 215–234. Springer, 1999.

[18] Rodolfo Dondo and Jaime Cerdá. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507, 2007.

[19] Edward Frazelle. World-class warehousing and material management, 2001.

[20] Chao Fu, Yiqiang Wang, Yan Gu, Mingyang Ma, and Tianyi Xue. Routing optimization of high-level orderpickers in a rectangular warehouse. In *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*, pages 4388–4391. IEEE, 2011.

[21] Noud Gademann and Steef Velde. Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE transactions*, 37(1):63–75, 2005.

[22] Marc Goetschalckx and H Donald Ratliff. Order picking in an aisle. *IIE transactions*, 20(1):53–62, 1988.

[23] Kevin R Gue, Russell D Meller, and Joseph D Skufca. The effects of pick density on order picking areas with narrow aisles. *IIE transactions*, 38(10):859–868, 2006.

[24] Ahmed Hassan and Bill Ferrell. Forklift routing in warehouses using dual-commands and stackable pallets. In *Proceedings of the International Material Handling Research Colloquium*, 2010.

[25] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[26] Ruhan He, Weibin Xu, Jiaxia Sun, and Bingqiao Zu. Balanced k-means algorithm for partitioning areas in large-scale vehicle routing problem. In *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, volume 3, pages 87–90. IEEE, 2009.

[27] Sebastian Henn, Sören Koch, and Gerhard Wäscher. Order batching in order picking warehouses: a survey of solution approaches. In *Warehousing in the global supply chain*, pages 105–137. Springer, 2012.

[28] DT Hiquebran, AS Alfa, JA Shapiro, and DH Gittoes. A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem. *Engineering Optimization*, 22(2):77–107, 1993.

[29] Matic Horvat. An approach to order picking optimization in warehouses. *Ljubljana: University of Ljubljana Faculty of computer and Information Science*, 2012.

[30] Jianyong Jin, Teodor Gabriel Crainic, and Arne Løkketangen. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*, 222(3):441–451, 2012.

[31] Ryan Key and Anurag Dasgupta. Warehouse pick path optimization algorithm analysis. In *11 th International Conference on Foundations of Computer Science (FCS 2015)*, 2015.

[32] Douglas M Lambert, James R Stock, and Lisa M Ellram. *Fundamentals of logistics management*. McGraw-Hill/Irwin, 1998.

[33] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358, 1992.

[34] Gilbert Laporte, Yves Nobert, and Serge Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation science*, 22(3):161–172, 1988.

[35] Anany Levitin. *Introduction to the design & analysis of algorithms*. Boston: Pearson,, 2012.

[36] Wenrong Lu, Duncan McFarlane, Vaggelis Giannikas, and Quan Zhang. An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, 248(1):107–122, 2016.

[37] Stephen Marsland. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2011.

[38] Christopher M Mohr. *Optimization of warehouse order-picking routes using vehicle routing model and genetic algorithm*. State University of New York at Binghamton, 2014.

[39] Rolf H Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speed up dijkstra's algorithm. In *International*

Workshop on Experimental and Efficient Algorithms, pages 189–202. Springer, 2005.

[40] Alwyn Jakobus Moolman, K Koen, and J Van der Westhuizen. Activity-based costing for vehicle routing problems. *South African Journal of Industrial Engineering*, 21(2):161–172, 2010.

[41] Gur Mosheiov. The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research*, 79(2):299–310, 1994.

[42] Phuong Khanh Nguyen, Teodor Gabriel Crainic, and Michel Toulouse. A tabu search for time-dependent multi-zone multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, 231(1):43–56, 2013.

[43] Charles G Petersen. An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, 17(11):1098–1111, 1997.

[44] H Donald Ratliff and Arnon S Rosenthal. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.

[45] Martin Reed, Aliki Yiannakou, and Roxanne Evering. An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, 15:169–176, 2014.

[46] Kees Jan Roodbergen and René De Koster. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43, 2001.

[47] Jim Rowley. *The Principles of Warehouse Design*. 01 2005.

[48] Catherine Gouveia Issac Jenne Sabrina Bant, Tristan Dowbiggin and Arshad Lasania. Fedex supply chain project, 2017.

[49] Qing Song and Xiaofan Wang. Partitioning graphs to speed up point-to-point shortest path computations. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 5299–5304. IEEE, 2011.

[50] Christophe Theys, Olli Bräysy, Wout Dullaert, and Birger Raa. Using a tsp heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763, 2010.

[51] James A Tompkins, John A White, YA Bozer, EH Frazelle, JMA Tanchoco, and J Trevino. Facility planning. *John Wiley & Sons, Inc., New York*, 1996.

[52] James A Tompkins, John A White, Yavuz A Bozer, and Jose Mario Azaña Tanchoco. *Facilities planning*. John Wiley & Sons, 2010.

[53] Paolo Toth and Daniele Vigo. An overview of vehicle routing problems. In *The vehicle routing problem*, pages 1–26. SIAM, 2002.

[54] Charles W Trigg. What is recreational mathematics? *Mathematics Magazine*, 51(1):18–21, 1978.

[55] University and Erasmus University Rotterdam the Erasmus School of Economics. Material Handling Forum. `https://www.erim.eur.nl/material-handling-forum/research-education/tools/calc-order-picking-time/what-to-do/short-explanation-on-layout/`, 2008. [Online; accessed 19-July-2008].

[56] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013.

[57] Gerhard Wäscher. Order picking: a survey of planning problems and methods. In *Supply chain management and reverse logistics*, pages 323–347. Springer, 2004.