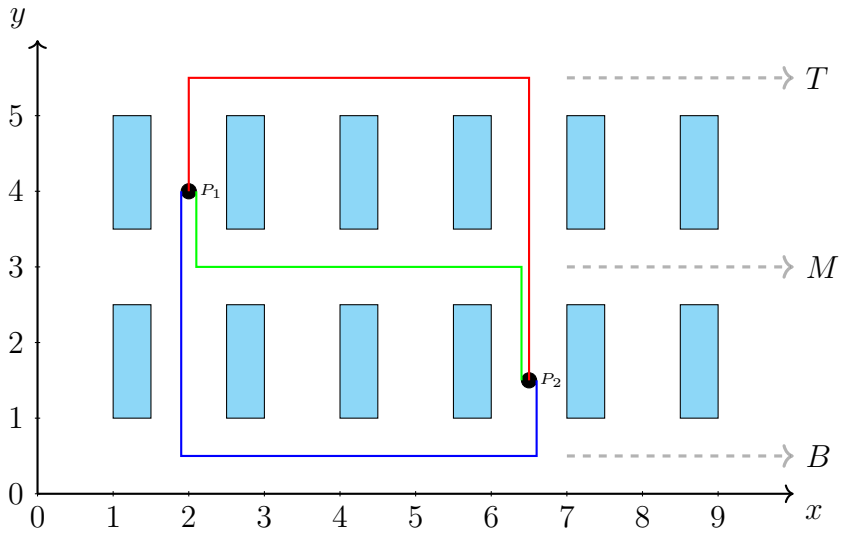


Optimizing routes for order picking in warehouses

A heuristic approach for the split delivery capacitated vehicle routing problem.

Master's thesis in Engineering Mathematics and Computational Science

LIRIM KADRIU

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Optimizing routes for order picking in warehouses

A heuristic approach for the split delivery capacitated vehicle routing problem

LIRIM KADRIU

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department Mathematical Sciences
Division of Mathematical Optimization
Chalmers University of Technology
Gothenburg, Sweden 2023

Optimizing routes for order picking in warehouses
A heuristic approach for the split delivery capacitated vehicle routing problem
LIRIM KADRIU

Supervisor: Sunney Fotedar, PhD student Chalmers University
Examiner: Ann-Brith Strömberg, Professor Chalmers University

Master's Thesis 2023
Department of Mathematics
Division of Mathematical Optimization
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Three possible Manhattan distances in a warehouse, when traversing between two shelf locations.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Abstract

The present study investigates the application of mathematical optimization in the context of warehouse logistics, with a particular focus on the Split Delivery Vehicle Routing Problem (SDVRP) encountered by the e-commerce company, Ellos Group. The SDVRP requires a solution that partitions demand between forklifts while satisfying routing and computational constraints in a warehouse setting where forklifts are used to travel and collect items from shelves, and items may vary in volume.

Since exact methods are time consuming, we propose a heuristic to identify good and feasible solutions for our optimization problem. Computational experiments based on practical (industrial) benchmark instances show that the proposed algorithm outperforms the existing routing policy method employed by Ellos in terms of solution quality. The results of this study suggest that the proposed algorithm has practical implications for improving the efficiency of warehouse operations and reducing transportation costs.

The study contributes to the field of mathematical optimization by providing a practical solution to a complex real-world problem in the context of warehouse logistics. The proposed heuristic's ability to solve the SDVRP with demand variability and routing constraints, along with constraints on computational time, demonstrates its efficacy in real-world settings. This work highlights the importance of mathematical optimization in modern industries and underlines the potential of heuristic methods in addressing complex real-world problems.

Keywords: Vehicle routing, Order Picking, Optimization, Heuristic, Warehouses.

Acknowledgements

This Master's thesis was conducted in collaboration with Ellos Group. Consequently, I would like to express my gratitude to Robert Tillander, Head of Advanced Analytics, and Matthias Parkhagen, Director of Supply Chain & Logistics, for making this thesis possible. I extend my appreciation to Niklas Venhagen, Logistics Developer, for his domain expertise and assistance in data collection. Additionally, I would like to thank my examiner, Professor Ann-Brith Strömberg, for her meticulous attention to mathematical rigor, which contributed to the quality of this work.

Foremost, I would like to express my sincere gratitude to Sunney Fotedar, my supervisor at Chalmers, for his unwavering support and insightful contributions throughout the duration of this project. I am profoundly thankful for the invaluable lessons he imparted on me, teaching me how to think critically and communicate effectively as a mathematician. Thank you, Sunney.

Lirim Kadriu, Gothenburg, June 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order.

LP	Linear Program
ILP	Integer Linear Program
MILP	Mixed-Integer Linear Program
MTZ	Miller-Tucker-Zemlin
PRP	Picker Routing Problem
TSP	Travelling Salesperson Problem
CVRP	Capacitated Vehicle Routing Problem
SDVRP	Split Delivery Vehicle Routing Problem
VNS	Variable Negiborhood Search
BVND	Basic Variable Neighborhood Descent

Contents

List of Acronyms **ix**

List of Figures **xii**

List of Tables **xiv**

1 Introduction **3**
1.1 Background . . . 3
1.2 Aim . . . 3
1.3 Outline . . . 4

2 Problem Description **5**
2.1 Warehouse layout . . . 5
2.2 Cost Matrix . . . 6
2.3 Limitations . . . 8

3 Theory **9**
3.1 Mathematical optimization . . . 9
3.2 Linear Programming . . . 10
3.3 Computational Complexity . . . 12
3.4 Network Graphs . . . 13
3.5 Routing Problems . . . 14
3.5.1 Traveling Salesperson Problems . . . 14
3.5.2 Vehicle Routing Problems . . . 15
3.5.3 The Split Delivery Vehicle Routing Problem . . . 15

4 Model **19**

5 Heuristic Approach **23**
5.1 A heuristic approach . . . 23
5.1.1 Route partitioning . . . 24
5.1.2 Local search : BVND . . . 25
5.1.3 Route Repair . . . 28

6 Tests and results **31**
6.1 Implementation and test settings . . . 31
6.1.1 Test results . . . 31

7 Discussion **35**

Bibliography **36**

List of Figures

1.1 *A typical distribution of an order picker's time. Source: [1]* 4

2.1 *A complete picture of the warehouse layout.* 6
2.2 *A figure illustrating the aggregation of two shelves to one coordinate point.* 7
2.3 *Illustration of three different possible Manhattan distances, from which the minimum is used as the distance between the locations P_1 and P_2.* 8

3.1 *A schematic overview of the optimization process. Source: SNYMAN, An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms* 10
3.2 *Difference between the optimal point in an ILP and after relaxing to an LP.* 11
3.3 *Different architectures for network graphs and their descriptions. Source: Wolfram Math World* 13
3.4 *Illustration of TSP solution.* 14
3.5 *Illustration of VRP solution. In this case, 3 vehicles are used to service all cities and each vehicle form a tour, blue, orange and red. All start and end at the same depot.* 15
3.6 *Illustration of a small-scale scenario where the VRP is compared to SDVRP. Both optimal solutions yield the same minimal cost, however the SDVRP uses 3 vehicles instead of 4. Source: [2]* 16
3.7 *In this scenario, the savings are greater as the savings do not only consist of one less vehicle but also a smaller total distance travelled. Source: [2]* 17

4.1 *The left subfigure shows a solution without having the subtour elimination constraints and the right subfigure shows a solution where the elimination constraints have been implemented.* 21
4.2 *Behavior of the variable u_{ik} in the MTZ.* 21

5.1 *Overview of the four components of the heuristic approach. Phase 3 is completed within phase 2.* 24
5.2 *Before swapping D and E* 26
5.3 *After swapping D and E* 26

6.1 *Bar plot of objective values at different steps in the heuristic approach including the model objective obtained from the Gurobi solver for the 180 min run. Observe that the model objective for problem instance F has been capped. The TSP objective is equivalent to the heuristic objective.* 33
6.2 *Bar plot of objective values for the S-shape picking policy currently used at Ellos Group compared to the Heuristic approach.* 34
6.3 *Bar plot indicating how many times one extra route was added out of 1000 simulations.* 34

List of Tables

2.1 *Key assumptions.* 8

4.1 *Table of notations.* 20

5.1 *Table of notations for the heuristic approach.* 24
5.2 *Compilation of the nine best combinations out of 72, tested in [3].* . . 25

6.1 *Problem instances and descriptions.* 31
6.2 *Results from using the SDVRP model using the Gurobi solver.* 32
6.3 *Results for the heuristic approach. The last column indicates the change in the objective value when comparing the objective value given by the Gurobi solver (180 minute run) in Table 6.2, to the objective value given by the heuristic approach. Negative values here correspond to lower heuristic objective value compared to the objective value given by the Gurobi solver.* 32
6.4 *Comparison of objective values at different steps in the heuristic. The Reduction-column is the reduction in objective value from the first route partition phase to the last TSP phase.* 33

1
Introduction

In order to maintain a competitive edge and sustain operations, businesses must prioritize constant improvement towards enhanced efficacy. In the realm of logistics, timely delivery and receipt of products is of significant importance. While external logistics pose challenges in terms of direct influence, internal logistics, particularly warehouse operations, offer potential opportunities for increased efficiencies.

1.1 Background

Ellos Group is a prominent e-commerce conglomerate in the Nordic region, comprising of several subsidiaries such as Ellos, Jotex, and Homeroom. As with many e-commerce firms, Ellos Group operates extensive warehouses that store a vast number of products, referred to as stock keeping units (SKUs). These SKUs are ordered by customers, necessitating their storage, retrieval, and shipping. Upon receiving a customer order, it is processed into a pick list, consisting of multiple SKUs that require collection, packaging, and sorting. Currently, this process relies on simple rule-based methods, lacking any formal optimization approach. However, Ellos Group has witnessed significant growth in recent years, leading to an expansion in picking operations. As a result, there is a need to explore formal optimization methods to enhance picking routines, which can guide pickers in navigating the warehouse, minimize costs, and enhance productivity.

In the context of optimizing and improving warehouse operations, it is often advisable to identify and prioritize steps or processes with the highest potential for reward relative to the associated effort. Of the various warehouse operations, such as receiving, cross-docking, put-away, order-picking, and shipping, order-picking is known to be the most resource-intensive and time-consuming. According to [1], order-picking involves retrieving SKUs from storage shelves to satisfy customer needs and typically accounts for 50%–70% of total operating costs in a typical warehouse. Therefore, significant gains in efficiency can be achieved by reducing the collective travel time associated with order-picking. Figure 1.1 illustrates the distribution of time costs across different warehouse operations.

1.2 Aim

The primary objective of this thesis is to propose a heuristic algorithm for optimizing pick route operations in warehouse logistics, where the demand comprises a heterogeneous set of items that vary in shape and volume. To achieve this objective,

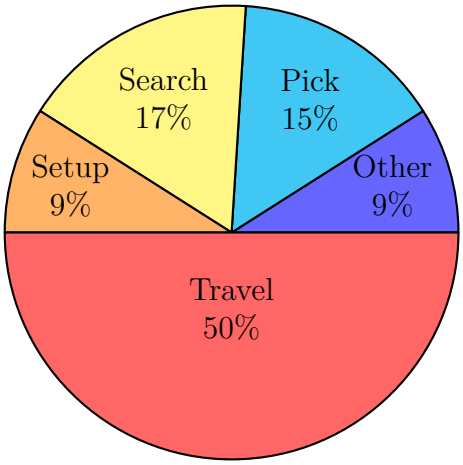


Figure 1.1: *A typical distribution of an order picker's time. Source: [1]*

the following factors will be taken into account:

1. Formulating a mathematical optimization model for routing forklifts in a warehouse while simultaneously considering various constraints.
2. Implementing the mathematical model to generate optimal pick routes.
3. Develop a heuristic to solve the problem quicker, albeit with a sacrifice of optimality.
4. Testing the heuristic against the model and the current picking policies at Ellos Group.

1.3 Outline

In Chapter 2, the problem description is formulated and the warehouse layout is presented. Moreover, the cost matrix is introduced as well as assumptions made and the limitations. Chapter 3 introduces the necessary theoretical tools used in this thesis. Next, Chapter 4 covers the formulation of the mathematical model used to solve the SDVRP. Chapter 5 contains the development of heuristic approach and how it is constructed using several sub-algorithms. Chapter 6 summarises the results in tables and visuals and finally Chapter 7 concludes with the discussion and closing thoughts. Chapter 7 also presents open questions and future work.

2
Problem Description

The daily picking operation in the warehouse setting is supported by the generation of new pick lists based on the previous day's customer orders. Before the initiation of the picking process, the demand and volume of each item are determined. The problem can be conceptualized as a combinatorial optimization problem, with a specific set of conditions and assumptions. This type of problem is also referred to as a picker routing problem (PRP) [4], where a picker, using a forklift, traverses the warehouse and collects items from shelves. The objective is to determine a route that visits all shelves and fulfills the demand while minimizing the travel time.

In Ellos, the daily picking operation is performed by a fleet of forklifts that simultaneously traverse the warehouse, each picking items from shelves. The dynamic nature of the warehouse implies that the location of items may vary due to supply chain delays or out-of-stock situations. Nonetheless, the location of each item is known at the start of each day's picking operation.

2.1 Warehouse layout

This study focuses on the analysis of a warehouse facility that serves as an order fulfillment center for Ellos Group AB, where items ordered by customers through multiple websites, are stored. The warehouse features a rectangular layout with equidistant aisles running parallel to the y-axis and cross-aisles located at the bottom, middle, and top sections of the facility, parallel to the x-axis. A comprehensive overview of the warehouse layout is provided in Figure 2.1. The depot, located in the bottom left corner of the facility referring to the coordinate $(0, 0)$, serves as the starting and ending point for all forklifts used in the warehouse. The blue rectangles depicted in the same figure represent the shelves where a specific type of item is stored. Each shelf stores multiple items, but only of the same type. In total, there are 3 cross-aisles located parallel to the x-axis and 34 aisles located parallel to the y-axis. The warehouse features 31 pick-shelves stacked vertically on top of one another, with a gap located in the mid cross-aisle. The three horizontal and vertical dots represent the continuation of the aisles. The aisles are positioned adjacent to one another, equidistant from each other, and accessible through the three cross-aisles located at the bottom, middle, and top sections of the facility. Despite being narrow, the aisles allow for U-turns and bi-directional travel.

In the remainder of this report, the term "pick location" denotes the coordinate point where a forklift stops when visiting a shelf, which is henceforth referred to as a "node". In order to simplify the problem, inner-aisle shelves in the warehouse

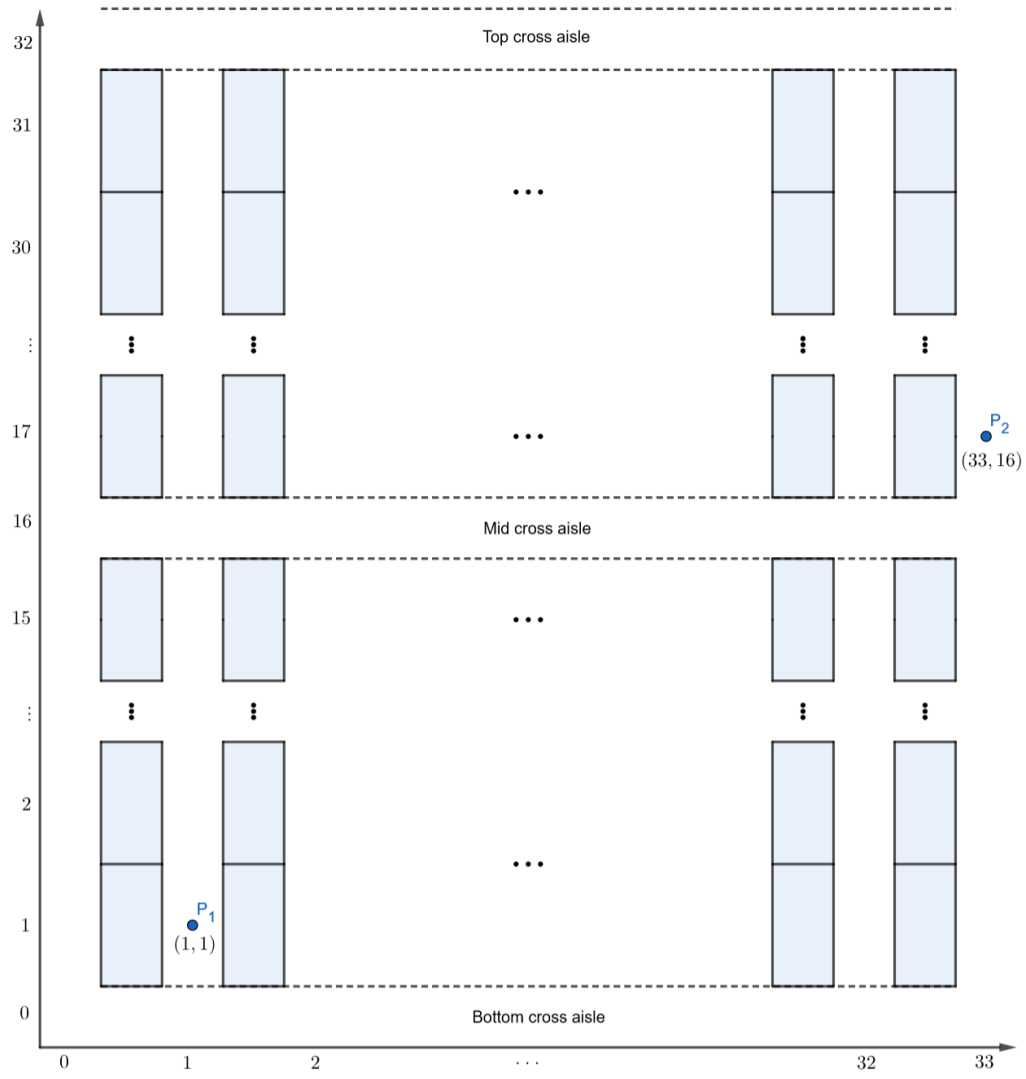


Figure 2.1: *A complete picture of the warehouse layout.*

layout are aggregated, such that two shelves correspond to a single node. This approach results in a nearly 50% reduction in the number of potential nodes. As an example, consider Figure 2.2, if a forklift visits either shelf 32 or shelf 63 it will stop at the same node regardless, at the node with coordinates $(1, 1)$. The dimensions and distances between shelves and aisles are also depicted in Figure 2.2, where the green nodes denote the actual coordinate points where forklifts stop while visiting shelves. The orange nodes do not point to any shelves and are located only on the cross-aisles. Finally, the red cross represents the location of the depot.

2.2 Cost Matrix

The mathematical model requires knowledge of the travel time. Let c_{ij} denote the travel time between shelf i and shelf j and note that the travel times are considered to be symmetric, i.e $c_{ij} = c_{ji}$. The distances calculated between each pair of shelves are pre-compiled in a cost matrix $\boldsymbol{C}$. Since the forklifts have to move around and

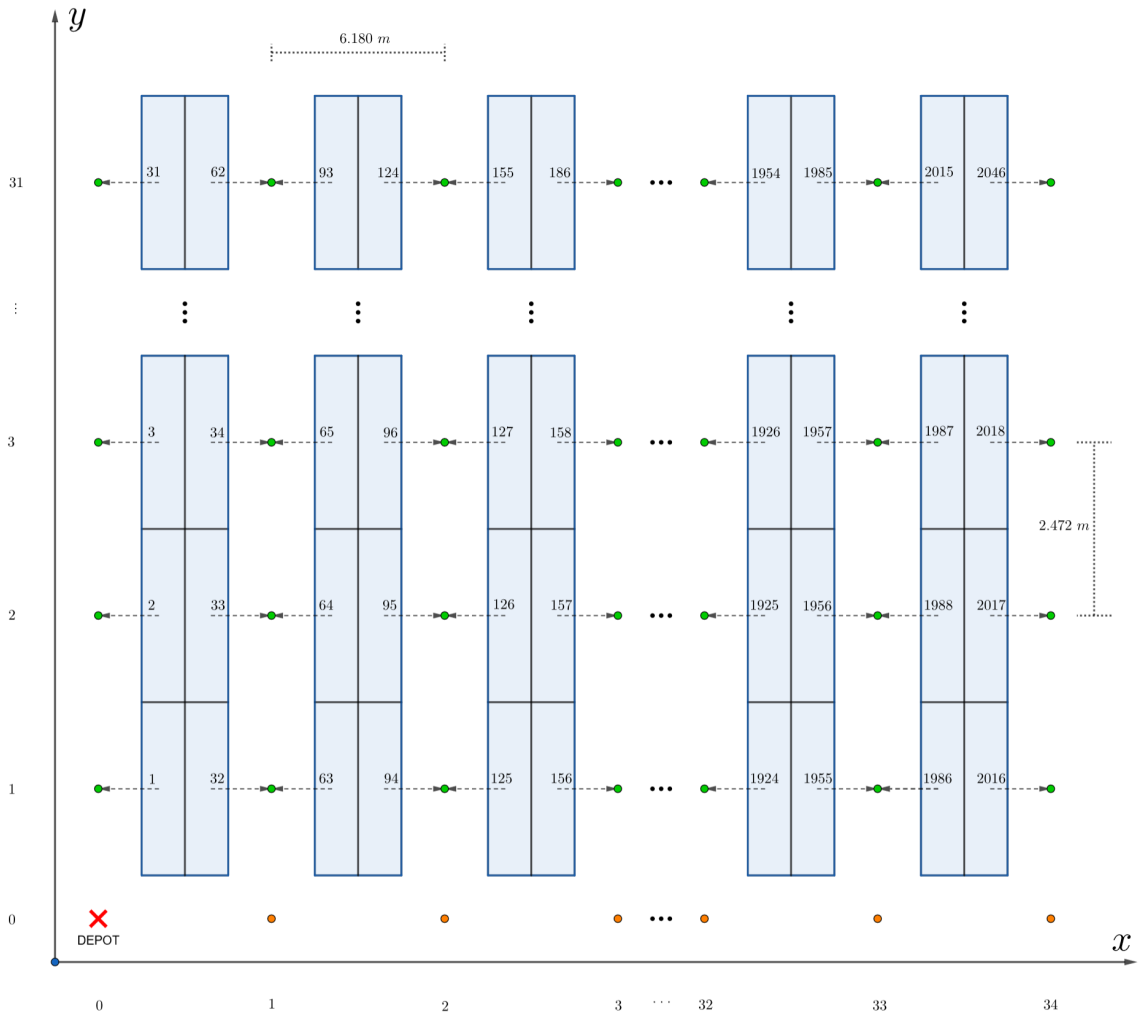


Figure 2.2: *A figure illustrating the aggregation of two shelves to one coordinate point.*

between racks of shelves, calculating the *Euclidean distance* will not be practical as it entails moving through the racks which is not realistic. In this case, a more appropriate distance measure is the *Manhattan distance* or more accurately, the minimum of three possible Manhattan distances as illustrated by Figure 2.3. This figure shows the different possibilities that a forklift can move, given the instruction to move from P_1 to P_2. The shortest distance between any pair of locations is calculated as follows. Let x be cross-aisle coordinate and y be the along-aisle coordinate as shown in Figure 2.3. If $x_i = x_j$ then the two items are located on the same aisle and the distance is computed as $c_{ij} = |y_j - y_i|$, else it is computed as:

$$\begin{aligned} \boldsymbol{C} \leftarrow c_{ij} = \min(&|y_i - B| + |x_j - x_i| + |y_j - B|, \\ &|y_i - M| + |x_j - x_i| + |y_j - M|, \\ &|y_i - T| + |x_j - x_i| + |y_j - T|), \qquad \forall \text{ shelves } i, j, \end{aligned} \tag{2.1}$$

where B, M, T are the y-coordinate of the bottom, middle and top cross-aisles, respectively.

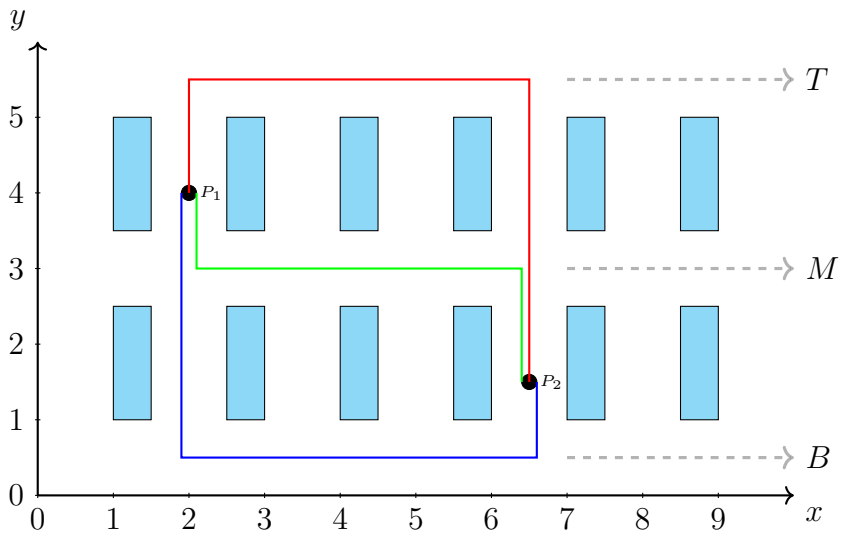


Figure 2.3: *Illustration of three different possible Manhattan distances, from which the minimum is used as the distance between the locations P_1 and P_2.*

2.3 Limitations

To simplify and accurately represent the system, we have made several assumptions and these are listed in Table 2.1. Notably, the inter-dependencies among forklifts cannot be completely addressed without a mechanism for each forklift to detect the position of the other forklifts. For instance, in a situation where a single forklift obstructs an aisle, other forklifts intending to traverse that aisle must either halt and wait or navigate an alternate path. To overcome this issue, the installation of RFID equipment on each forklift is needed in order to transmit its location like a beacon. Furthermore, the restriction that prohibits two forklifts from reaching the same location at the same time is not accounted for in the optimization model. As a consequence, the optimal routes computed may in fact turn out to be suboptimal, if conditions are unfortunate.

Table 2.1: *Key assumptions.*

#	**Assumptions**
1.	There are no obstacles on the travel route
2.	There is no interaction between forklifts
3.	All forklifts have the same volume capacity
4.	The locations of each shelf are known
5.	The forklifts travel at constant speed
6.	The problem is only considered in two dimensions
7.	All items can be stacked when the forklifts are loaded
8.	The aisles are bidirectional and U-turns are allowed
9.	The shortest route between two shelves is known
10.	Each shelf contains one unique type of item

3

Theory

The current chapter expound the theoretical underpinnings of the methods utilized to formulate the pick route optimization problem and the associated heuristic approach. The primary emphasis of this discussion is placed upon routing problems, given their essential role in establishing the groundwork for effective pick routing. Presenting formal proofs has been refrained from, as these have been previously established within referred literature.

3.1 Mathematical optimization

Optimization refers to the process of achieving the best possible outcome in a given situation, given the existing constraints. The term "optimum" has its roots in the Latin language and signifies the "ultimate deal". In order to attain such optimal outcomes, it is necessary to mathematically model real-world problems and devise appropriate solutions. One classic example of a problem that necessitates mathematical optimization is portfolio optimization, wherein an investor seeks to determine the optimal allocation of available capital across a range of assets to maximize return. The constraints may include minimum return, budgetary limitations, and minimum/maximum investment per asset. Another example pertains to the optimization of athlete performance, where determining the optimal consumption levels of macronutrients is critical for achieving optimal performance. The constraints in this case may include minimum/maximum intake limits for each nutrient and a ceiling on total caloric intake. In practice, solving such problems is often challenging, particularly when dealing with large problem instances featuring numerous constraints or variables (decisions). Optimization engineers often confront trade-offs between computational efficiency and solution quality. A general mathematical formulation of an optimization problem takes the following form:

$$\begin{aligned}
\min \quad & f(\boldsymbol{x}) && && \text{(3.1a)}\\
\textbf{s.t.} \quad & g_j(\boldsymbol{x}) \leq 0, && j = 1, \ldots, m, && \text{(3.1b)}\\
& h_j(\boldsymbol{x}) = 0, && j = 1, \ldots, r, && \text{(3.1c)}\\
& \boldsymbol{x} \in \mathbb{R}^n && && \text{(3.1d)}
\end{aligned}$$

where $f(\boldsymbol{x})$ is the value of the function $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g_j(\boldsymbol{x})$ and $h_j(\boldsymbol{x})$ are scalar functions of the column vector $\boldsymbol{x}$, i.e. $g, h : \mathbb{R}^n \mapsto \mathbb{R}$. The continuous components x_i of $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$ are called the *design/decision variables*, f is the *objective function*, i.e. the function we intended to optimize, g_j denotes the *inequality constraint*

function and h_j the *equality constraint function*. The optimal n-dimensional vector $\boldsymbol{x}$ that solves the problem is denoted by $\boldsymbol{x}^*$ with corresponding function value $f(\boldsymbol{x}^*)$. If there are no $h_j(\boldsymbol{x}) \leq 0$ and $g_j(\boldsymbol{x}) \leq 0$, the problem is referred to as *unconstrained optimization problem*. It is worth noting that a maximization problem is easily translated into a minimization problem by observing that $\max_{\boldsymbol{x}} f(\boldsymbol{x}) = -\min_{\boldsymbol{x}}\{-f(\boldsymbol{x})\}$.

Mathematical modelling is considered to be a process consisting of several phases. Normally, there is a question that needs a decision to be made and from here, a real problem is identified. For example, if a FedEx delivery vehicle needs to deliver 10 customer orders to different locations in a city and the customers can only receive the orders at certain time windows during the day, how should the vehicle operator plan his route through the city in order to minimize the distance travelled, hence minimizing fuel cost? Once the problem has been identified, a mathematical formulation of the optimization problem is needed, which is then implemented and solved using an appropriate software and solver. Finally, the model must be evaluated and verified.

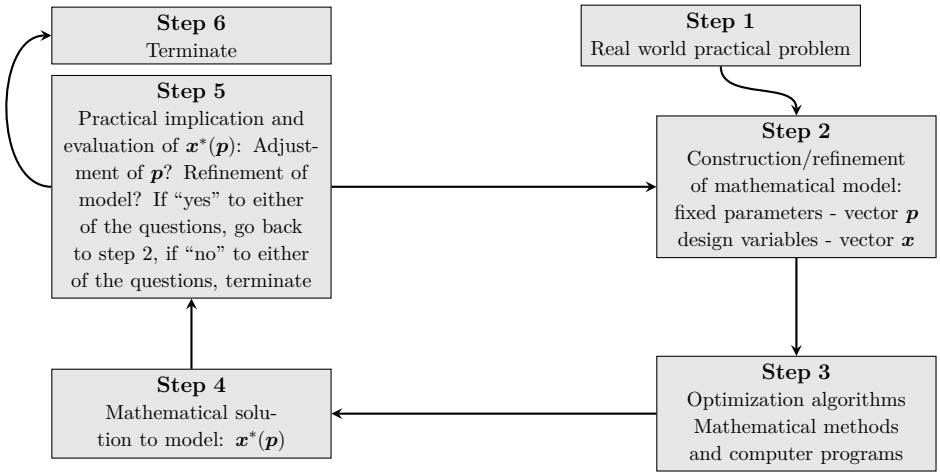


Figure 3.1: *A schematic overview of the optimization process. Source: SNYMAN, An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*

3.2 Linear Programming

When the constraints and objective functions in an optimization problem are affine functions of $\boldsymbol{x} \in \mathbb{R}^n$ the optimization problem is then called an *Linear Programming* (LP). In the two dimensional case, the set of feasible solutions form a polygon $\mathcal{S}$ defined by the intersection of finitely many half-spaces generated by the inequality constraints. The objective of linear programming algorithm is to find points on this polygon that yield the smallest or largest function value. If a LP has an optimal solution, then that solution will also be a vertex in the polyhedron.

There are extensions of LP to *Integer Linear Programming* (ILP) and also *Mixed Integer Linear Programming* (MILP). For a LP to become a ILP, the inte-

grality constraint $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{Z}^n$ is added. Conversely, the removal of the integrality constraint is called *relaxation* of a ILP to a LP. By imposing the integrality constraint, the optimal point no longer needs to coincide with a vertex of the polygon of the LP relaxation as Figure 3.2 shows. The set of ILP solutions is a subset of the set of LP solutions. To obtain good approximations of an optimal solution to MILPs, integer relaxations of MILPs are widely used since LPs are generally easier to solve, see next section.

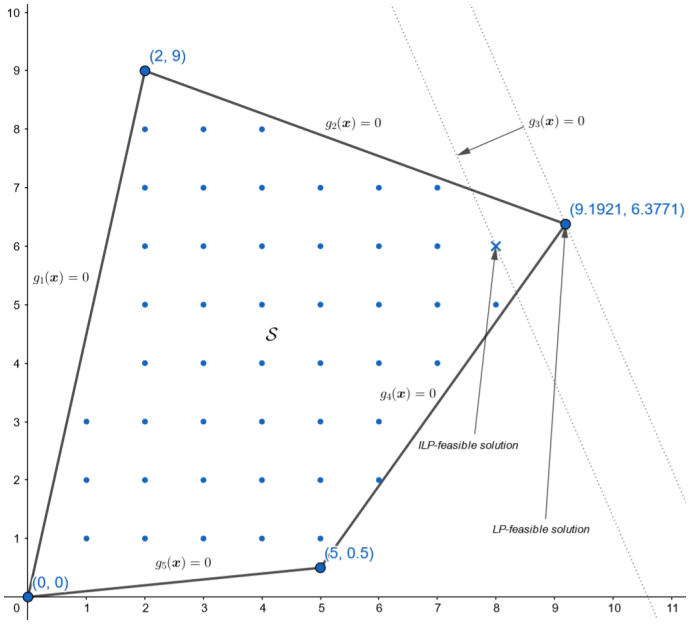


Figure 3.2: *Difference between the optimal point in an ILP and after relaxing to an LP.*

The core problem in this thesis is a MILP and these are generally of the form

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{h}^\top \boldsymbol{y} && \text{(3.2a)} \\ \textbf{s.t.} \quad & A\boldsymbol{x} + G\boldsymbol{y} \leq b && \text{(3.2b)} \\ & \boldsymbol{x} \in \mathbb{Z}_+^{n_1}, && \text{(3.2c)} \\ & \boldsymbol{y} \in \mathbb{R}_+^{n_2}, && \text{(3.2d)} \end{aligned}$$

where the parameters are assumed to be rational, denoted by $\boldsymbol{c} \in \mathbb{Q}_+^{n_1}, \boldsymbol{h} \in \mathbb{Q}_+^{n_2}, A \in \mathbb{R}^{m \times n_1}, G \in \mathbb{R}^{m \times n_2}$. The variable $\boldsymbol{x}$ is non-negative and integral and $\boldsymbol{y}$ is non-negative and continuous. The feasible set to (3.2) is denoted

$$\mathcal{S} := \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z}_+^{n_1} \times \mathbb{R}_+^{n_2} \mid A\boldsymbol{x} + G\boldsymbol{y} \leq b\}. \tag{3.3}$$

An integer relaxation of (3.2) is

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{h}^\top \boldsymbol{y} && \text{(3.4a)} \\ \textbf{s.t.} \quad & A\boldsymbol{x} + G\boldsymbol{y} \leq b && \text{(3.4b)} \\ & \boldsymbol{x} \in \mathbb{R}_+^{n_1}, && \text{(3.4c)} \\ & \boldsymbol{y} \in \mathbb{R}_+^{n_2}, && \text{(3.4d)} \end{aligned}$$

with corresponding feasible set

$$\mathcal{S}_{\text{Relaxed}} := \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}_+^{n_1} \times \mathbb{R}_+^{n_2} \mid A\boldsymbol{x} + G\boldsymbol{y} \leq b\}. \tag{3.5}$$

There are many different solution methods for solving MILPs, many of these are heuristics and stochastic methods that don't provide exact solutions. In this thesis we employ the *Branch-and-bound* algorithm which operates on the "divide and conquer" principle in which two or more sub-problems are created by restricting the domain of a variable or a group of variables [5, Chapter 3, Section 3.1.1]. We begin with the original MILP we intend to solve but we don't know how to solve this problem directly, so we relax the problem into an LP and now we can use for example the *Simplex* method [6] to solve the relaxed problem. At this point, one of two things can happen:

1. If we are lucky, the solution to the resulting LP happens to satisfy all the integrality constraints without these being explicitly imposed in the formulation, then we are done. The optimal solution to this LP is also the optimal solution to corresponding original MILP.

2. As is usually the case in most real world problems, if the solution to the LP is rarely the solution to to original MILP, then we divide the master problem into sub-problems. As an example, suppose that $x = 9.3$ is the solution to the LP. To exclude this value we simply impose the two constraints $x \leq 9$ and $x \geq 10$. Thus, if the original MILP is denoted P_0 then the two new sub-problems can be denoted P_1, P_2 where the constraints $x \leq 9, x \geq 10$ are imposed respectively. If we can compute optimal solutions for both P_1 and P_2, we choose the better solution of the two sub-problems and it will be the optimal solution to the original MILP P_0. If we can't compute the optimal solutions to the two sub-problems, we continue the branching and repeat this process until we are successful. This procedure will generate a *search tree* where all nodes are sub-problems $P_1, ..., P_i, i \in \mathbb{N}_+$. The leaves of this tree are nodes that are yet to be branched.

The *bounding* part of the branch-and-bound algorithm is done by solving the LP-relaxations of generated sub-problems. The best objective value found at any point in the search is referred to as the *upper bound*. There is also a *lower bound* at any point in the search that is computed by taking the minimum of all the optimal objective values of the current leaf nodes. Finally, the *gap* is defined as the difference between these two bounds. Reaching $GAP = 0$ indicates we have reached optimality.

3.3 Computational Complexity

Computational complexity theory is a field of study in which the focus is to classify different computational problems based on resource consumption, such as time and memory. A problem is considered to be *difficult* to solve if the resources it requires grows exponentially as the size of the instance increases. The problem of finding the

shortest path in a graph given various constraints is considered to be a computational problem.

The two classes of problems are P and NP. P stands for "polynomial time" and refers to problems for which a solution can be found in a time that is polynomial in the size of the input. These problems are considered to be "easy" to solve. Examples of P problems include sorting and searching algorithms, such as binary search. NP, on the other hand, stands for "nondeterministic polynomial time" and refers to problems for which it is not known whether a solution can be found in polynomial time. However, if a solution is found, it can be verified in polynomial time. These problems are considered to be "hard" to solve. It is still an open question whether $P = NP$, meaning whether all NP problems can be solved in polynomial time or not. Generally, MILPs are NP-hard [7, Chapter 1.3].

3.4 Network Graphs

A *graph network* is simply a set of nodes (vertices) and edges that are connected in certain ways. Depending on the ways the nodes are connected, the graphs are grouped into different categories, as Figure 3.3 shows. The set of vertices are usually denoted as $\mathcal{N} = \{n_1, n_2, ..., n_m\}, m \in \mathbb{N}_+$ and the set of edges are denoted $\mathcal{E} = \{e_1, e_2, ..., e_p\}, p \in \mathbb{N}_+$, together they form the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. An edge e_i is expressed as an unordered pair (n_j, n_k), where $n_j, n_k \in \mathcal{N}$. In the particular case where $j = k$, the node n_j has a connection with it self and a *loop* is formed. A graph is said to be *complete* if all pairs of nodes are connected, i.e have a direct edge between them.

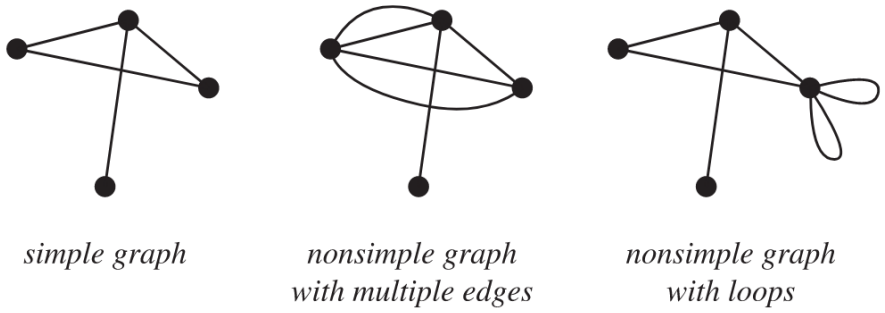


Figure 3.3: *Different architectures for network graphs and their descriptions. Source: Wolfram Math World*

If we were to start at a certain node and traverse through a certain number of nodes, never pass the same node that we already passed and finally return to the starting node, it is said that we have completed a *cycle*. A special cycle is the *Hamiltonian cycle*. This is a graph cycle that is a closed loop through a graph, in which each node is visited exactly once. If a graph possesses a Hamiltonian cycle it is said to be a *Hamiltonian graph*.

3.5 Routing Problems

Routing problems is a category of optimization problems that aims to find optimal routes to delivery or pick up locations. In numerous cases, the objective is to transport goods to or from customers/locations as efficiently as possible. For instance, postal workers may need to deliver packages to customers, while warehouse pickers may need to retrieve items from various shelves. Routing problems encompass a wide range of variations, but those that are relevant to the routing problem investigated in this study are explicated below.

3.5.1 Traveling Salesperson Problems

The problem of finding the minimum cost of a Hamiltonian cycle is called *Traveling Salesperson Problem (TSP)*, which is a famous problem with somewhat obscure origins; it was mentioned in an 1832 manual for traveling Salesperson but with no mathematical consideration [8]. W. R. Hamilton and Thomas Kirkman devised mathematical formulations of the problem in the 1800's but it was not until the 1930's when its general form was studied by Karl Menger in Vienna and Harvard [9]. Its formulation is simple to convey and understand: *Given a set of cities and the distances between each pair of cities, what is the shortest possible route the Salesperson can take such that each city is visited exactly once and return to his original city?* More formally it is described as follows. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a directed or undirected graph with a set of vertices $\mathcal{N}$ and a set of edges $\mathcal{E} = \{(x, y) \ : \ x, y \in \mathcal{N}\}$. Each edge $e \in \mathcal{E}$ is assigned a cost c_e. Let $\mathcal{H}$ be set of all Hamiltonian cycles in $\mathcal{G}$, the traveling Salesperson problem entails finding the tour $h \in \mathcal{H}$ such that $\sum_{e \in \mathcal{E}} c_e$ is minimized [10, 11].

Given a starting node (or city), it is easy to calculate the number of different paths to n nodes. If the Salesperson is currently in one of the nodes $n \in \mathcal{N}$, there are $n - 1$ ways to choose the next node to visit, after which there are $n - 2$ nodes left to visit and so on. Multiplying all of these choices gives $(n - 1)!$ different paths the Salesperson could take. Since the cost matrix is symmetric, the total number of different paths is divided by 2 and we arrive at $(n - 1)!/2$. The number of possible solutions grows exponentially as a function of n. A simple solution for the case $n = 7$ is illustrated by Figure 3.4.

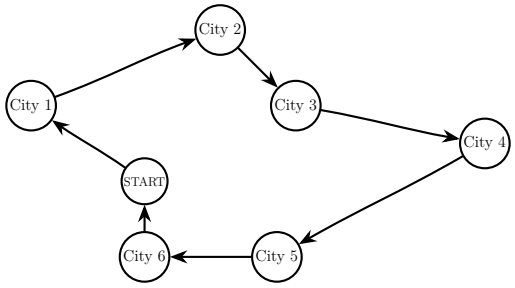


Figure 3.4: *Illustration of TSP solution.*

3.5.2 Vehicle Routing Problems

The Vehicle Routing Problem is an extension of the Traveling Salesperson Problem in the sense that instead of one Salesperson, the VRP considers several salesmen that visit the given locations. It was first introduced by as truck dispatching problem by Dantzig and Ramser in [12]. All cities must be visited exactly once and the demand of goods at each city must be fulfilled. Moreover, several constraints are added depending on the type of problem considered, such as the realistic assumption that each vehicle has a finite capacity, a certain city can only be visited within a time window, the cities must be visited in specific order etc. Figure 3.5 illustrates a potential VRP solution. The VRP and TSP are both $NP-$hard problems.

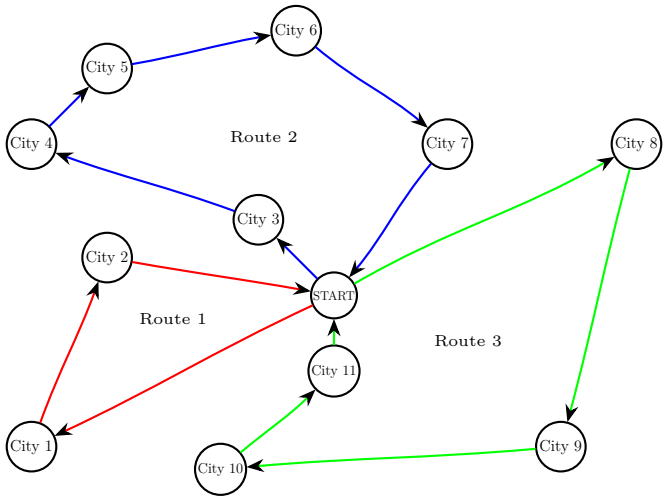


Figure 3.5: *Illustration of VRP solution. In this case, 3 vehicles are used to service all cities and each vehicle form a tour, blue, orange and red. All start and end at the same depot.*

3.5.3 The Split Delivery Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) is a widely studied problem in the field of transportation logistics. However, the basic CVRP formulation assumes that each customer can only be serviced by a single vehicle, which often results in underutilization of available resources. To address this limitation, an alternative formulation called the Split Delivery Vehicle Routing Problem (SDVRP) has been proposed. The SDVRP allows for a customer to be serviced by multiple vehicles, thereby increasing the utilization of vehicles and reducing overall transportation costs.

The SDVRP was first proposed in [13] where the discussion revolved around a customer delivery problem with delivery splitting. [13, 14] have analyzed the savings generated by allowing split deliveries in a VRP. They showed that when the distances satisfy the triangle inequality, there exists an optimal solution for the SDVRP where no pair of tours has two or more customers in common. A comparison between SDVRP and traditional VRP has been made in [14] which concluded that splitting the deliveries rather than allowing a customer demand to be served by only

one vehicle, is more effective since it usually reduces the number of vehicles required and hence the total distance travelled. If the demands at each location exceeded 10% of vehicles capacity, which is often the case in warehouses, very significant cost savings are realized when allowing split deliveries according [14](p. 394–400). Many times the SDVRP takes more computer time to solve than the VRP since the feasible space is larger [14, 2, 15].

Take Figure 3.6 as an example graph system where a comparison between SDVRP and VRP is made. Assume that each vehicle is only able to carry four units.

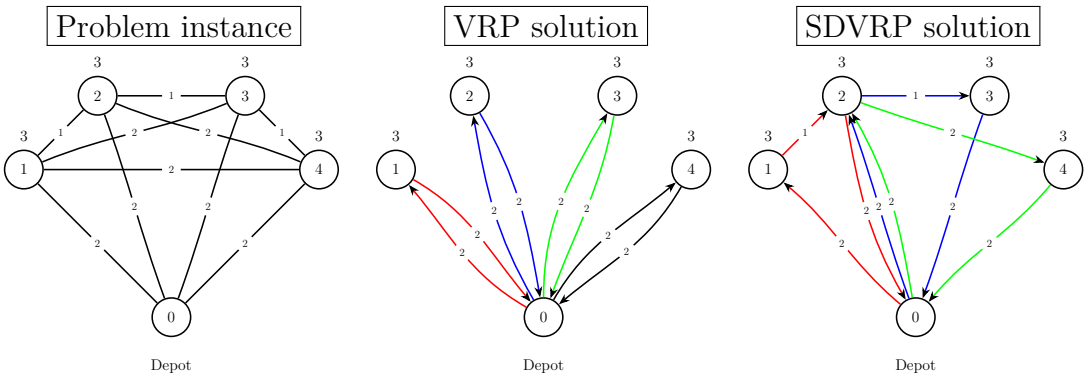


Figure 3.6: *Illustration of a small-scale scenario where the VRP is compared to SDVRP. Both optimal solutions yield the same minimal cost, however the SDVRP uses 3 vehicles instead of 4. Source: [2]*

The left sub-figure shows the depot, the four customers where each has a demand of three units, and the arcs between each pair of locations along with their associated costs. The center sub-figure shows the optimal VRP solution with four vehicles, each going to one customer and back. The total cost is the sum of all used arcs, which in this case is 16. The right figure shows the optimal SDVRP solution on the same graph, where only three vehicles are used to complete the operation. The total cost in this case is also 16, however one vehicle is saved.

It is also interesting to see if there exist cases where the optimal solution of the SDVRP has a lesser cost than the cost of the optimal solution to the corresponding VRP. Figure 3.7 shows such an instance where the vehicle capacity is set to be 5. The solution of the VRP has a minimum cost of 24, whereas the solution for the SDVRP has a solution with minimum cost of 18. Note that the solution of the SDVRP also uses two vehicles as opposed to the solution of the VRP which uses three vehicles. Hence, savings can be achieved both in terms of number of vehicles and the distance travelled.

In general, in the SDVRP, a solution always exists that makes use of the minimum possible number of vehicles [2]. This number can be easily computed as it is the ratio between the overall demand and the vehicle capacity, rounded to the up nearest integer. A better solution, in terms of the minimum total cost of the routes, may, however, exist that makes use of a larger number of vehicles.

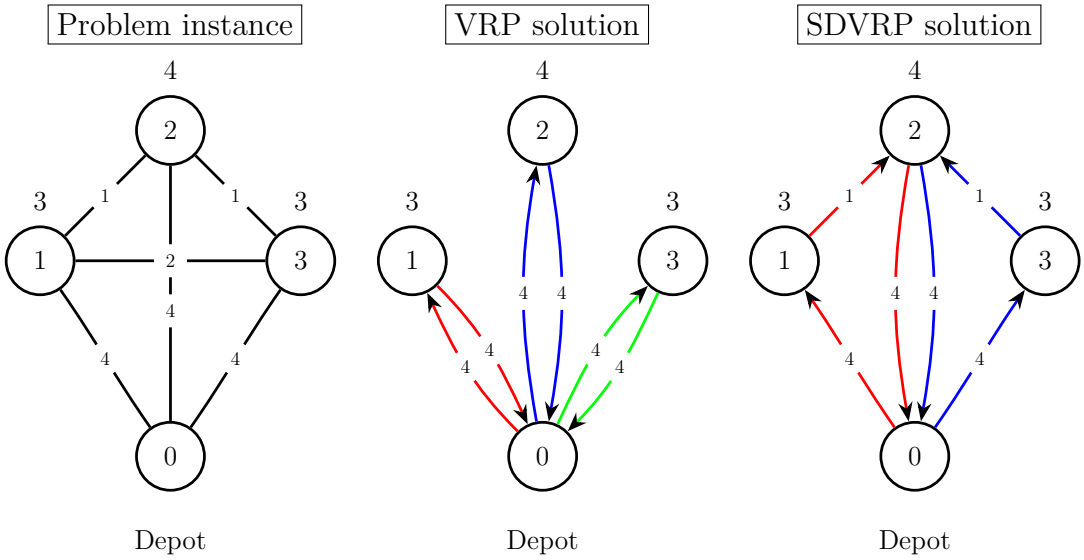


Figure 3.7: *In this scenario, the savings are greater as the savings do not only consist of one less vehicle but also a smaller total distance travelled. Source: [2]*

4
Model

We consider the SDVRP formulation where a fleet of homogeneous forklifts have to collect items from a subset of shelves in Ellos warehouse. In contrast to what is usually assumed in classical VRP's, the demand at each shelf *can* be split among forklifts, meaning that each shelf can be visited more than once. However, despite this relaxation, the SDVRP remains computationally hard to solve [16, Introduction]. Moreover, the demand at one location can be greater than the maximum capacity of a forklift. There is a depot where each forklift has to start and end its route. The objective is to find a set of forklift routes that collect items at all the shelves with positive demand, such that the sum of the quantities picked in each tour does not exceed the capacity of any of the forklifts and such that the total distance traversed by the forklift fleet is minimized. In this setting, *vehicle* is replaced by *forklift*, *deliver* is replaced by *pick/collect* and *customer* is replaced by *shelf*. We also make the distinction between *item-demand* and *volume-demand*. The former is used to describe the total number of items of a certain type to be picked and the latter describes the total volume in dm^3 that said items entail.

The fleet of forklifts is given and denoted by the set $\mathcal{K}$ and the set $\mathcal{N}$ denotes the set of shelves to be visited, i.e. the daily pick list. The notation $\mathcal{N}_0$ represents the set of shelves including the depot, i.e. $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$. For each shelf $i \in \mathcal{N}$ there is an item-demand d_i and a volume v_i associated with the item residing on the i^{th} shelf. The cardinalities of the sets $\mathcal{K}$ and $\mathcal{N}$ are $K_{\min}$ and N, respectively. The $K_{\min}$ is used in order to guarantee that a feasible solution exists and is elaborated in the second last paragraph of this chapter. All notations are described in Table 4.1.

The model (4.1) is a modification of the original SDVRP as proposed by [13] including capacity constraints on the forklifts in terms of volume. Furthermore, the subtour elimination constraints (4.1c) use the Miller-Tucker-Zemlin (MTZ) formulation as opposed to the Dantzig-Fulkerson-Johnson (DFJ) formulation proposed by [13, 14].

Table 4.1: *Table of notations.*

Notation	Explanation
$\mathcal{K} = \{1, ..., K_{\min}\}$	Set of forklifts traversing the warehouse
$\mathcal{N} = \{1, ..., N\}$	Set of shelves on the picklist, excluding depot
$\mathcal{N}_0 = \mathcal{N} \cup \{0\} = \{0, ..., N\}$	Set of shelves on the picklist, including depot
$\mathcal{E} = \{(i,j) \mid i,j \in \mathcal{N}_0 \times \mathcal{N}_0\}$	Set of feasible paths between shelves
$i, j \in \mathcal{N}_0$	Indices for shelves
$k \in \mathcal{K}$	Index of the forklifts
V	Maximum volume capacity of one forklift
c_{ij}	Cost of shortest path between shelves i and j
d_i	Demand at shelf i
v_i	Volume for on unit of the item associated with shelf i
$D = \sum_{i \in \mathcal{N}} d_i v_i$	Total volume-demand of picklist
$M > \sum_{i \in \mathcal{N}} d_i$	An arbitrary large number used in the MTZ constraint
$x_{ijk} \in \mathbb{B}$	Is 1 if forklift k uses arc $(i,j) \in \mathcal{E}$ and visits shelf j, 0 otherwise
$y_{ik} \in \mathbb{Z}_+$	Number of items from the i^{th} shelf picked by the k^{th} forklift
$u_{ik} \in [0, M]$	Dummy variables for the MTZ sub-tour elimination. These extra variables keep track of the ordering of the nodes to prevent illegal subtours

The mathematical formulation is as follows:

$$\text{minimize} \quad f(\boldsymbol{x}) = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}_0} \sum_{j \in \mathcal{N}_0} c_{ij} x_{ijk} \tag{4.1a}$$

subject to

$$\sum_{\substack{i \in \mathcal{N}_0 \\ i \neq p}} x_{ipk} - \sum_{\substack{j \in \mathcal{N}_0 \\ j \neq p}} x_{pjk} = 0, \qquad p \in \mathcal{N}_0, \quad k \in \mathcal{K}, \tag{4.1b}$$

$$u_{ik} + y_{ik} - M(1 - x_{ijk}) \leq u_{jk}, \qquad (i,j) \in \mathcal{E}, \quad k \in \mathcal{K}, \tag{4.1c}$$

$$\sum_{\substack{j \in \mathcal{N}_0 \\ j \neq i}} d_i x_{ijk} \geq y_{ik}, \qquad i \in \mathcal{N}, \quad k \in \mathcal{K}, \tag{4.1d}$$

$$\sum_{k \in \mathcal{K}} y_{ik} = d_i, \qquad i \in \mathcal{N}, \tag{4.1e}$$

$$\sum_{i \in \mathcal{N}} y_{ik} v_i \leq V, \qquad k \in \mathcal{K}, \tag{4.1f}$$

$$x_{ijk} \in \mathbb{B}, \qquad (i,j) \in \mathcal{E}, \quad k \in \mathcal{K}, \tag{4.1g}$$

$$y_{ik} \in \mathbb{Z}_+, \qquad i \in \mathcal{N}, \quad k \in \mathcal{K}, \tag{4.1h}$$

$$y_{ik} \leq u_{ik} \leq M, \qquad i \in \mathcal{N}, \quad k \in \mathcal{K}, \tag{4.1i}$$

where (4.1a) is the objective function representing the total distance travelled during

the pick operation, (4.1b) are the classical routing constraints stating that if a forklift visits a shelf, it must also leave it. Constraints (4.1c) are used for eliminating subtours and is discussed in the next paragraph, (4.1d) does not allow items to be picked unless the shelf is visited. Constraints (4.1e) enforce that the total quantity collected by all forklifts picking from shelf i is equal to the total item-demand at shelf i. Constraints (4.1f) enforces that the maximum volume capacity of the forklifts are not violated and constraints (4.1g)–(4.1i) specify the decision variables.

If constraints (4.1c) were ignored, we would get subtours. These are tours that lack a direct path between each other. Figure 4.1 shows a solution for one forklift. In the left subfigure there are two subtours and no way to move between these.

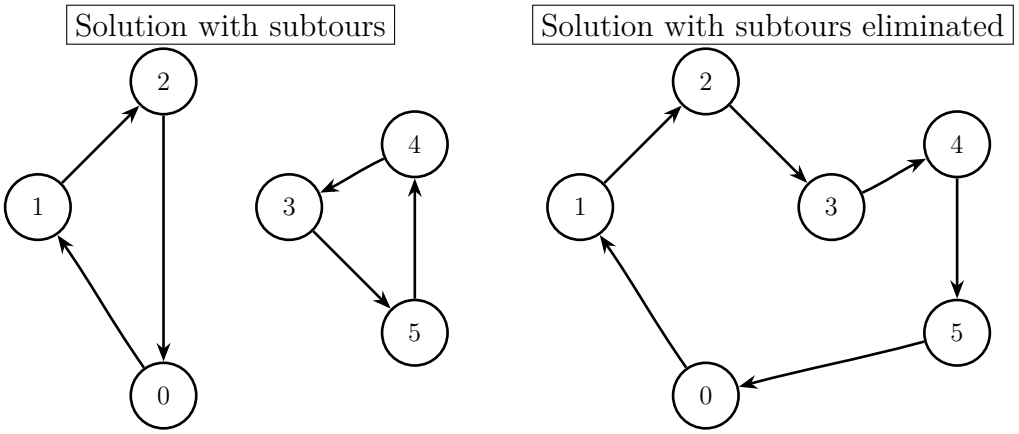


Figure 4.1: *The left subfigure shows a solution without having the subtour elimination constraints and the right subfigure shows a solution where the elimination constraints have been implemented.*

The subtour elimination constraints (4.1c) is known as the Miller-Tucker-Zemlin (MTZ) formulation [17]. This formulation uses extra integer variables $u_{ik}, i \in \mathcal{N}$ which obtain a value for each shelf in $\mathcal{N}$. If a forklift k travels from shelf i to shelf j then $u_{jk} > u_{ik}$, hence each time a new shelf is visited the value of u_{ik} increases by one.

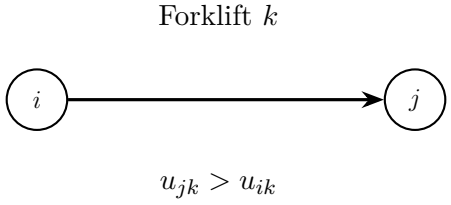


Figure 4.2: *Behavior of the variable u_{ik} in the MTZ.*

Looking at the constraint (4.1c), the number M is some arbitrary large number such that

$$M > \sum_{i \in \mathcal{N}} d_i. \tag{4.2}$$

If forklift k travels directly from i to j then $x_{ijk} = 1$ and the constraint can be more simply expressed as $u_{ik} + y_{ik} \leq u_{jk}$ which implies that the cumulative load at shelf j in forklift k is at least y_{ik} higher than u_{ik}, where y_{ik} is a fraction of the total item-demand d_i picked by forklift k.

If on the other hand forklift k does not travel from i to j then $x_{ijk} = 0$ and the inequality is expressed as $u_{ik} + y_{ik} - M \leq u_{jk}$. Moreover, since $M \gg u_{ik} + y_{ik}$ we have that $u_{jk} \geq -M$ which implies a trivial constraint as it is in agreement with (4.1i) stating that $M \geq u_{jk} \geq 0$.

The mathematical model needs a lower bound on the number of forklifts required. The quantity picked in each tour can not exceed the maximum volume capacity $V_k = V, \ \forall k \in \mathcal{K}$. Since each item also has a certain volume associated with it, the item-demand quantity d_i of an item must be multiplied with its volume v_i to obtain the volume-demand. The lower bound for the number of forklifts is thus calculated by

$$K_{\min} = \left\lceil \frac{1}{V} \sum_{i \in \mathcal{N}} d_i v_i \right\rceil = \left\lceil \frac{D}{V} \right\rceil. \tag{4.3}$$

In [15] it was shown that there always exists an optimal solution that uses at most $K_{\min}$ forklifts.

Upon implementing and testing the mathematical model (4.1), it is found that obtaining good solutions takes long time for solving large picking problems. The computational complexity makes this formulation impractical for daily use, hence we device a heuristic approach, introduced in the next chapter.

5
Heuristic Approach

This chapter outlines a heuristic approach that has been developed to address the computational costs of solving the SDVRP formulation (4.1). The approach is based on a route partitioning procedure, which quickly generates feasible routes for each forklift. The routes are then improved by means of a local search method. Finally, a TSP solver is utilized to determine the optimal order of visits to the shelves for each route. The various components and steps of the heuristic algorithm are elaborated in detail in this chapter.

5.1 A heuristic approach

The proposed heuristic approach is comprised of four sequential phases, whereby the output of the preceding phase serves as the input for the subsequent phase. The initial phase, referred to as the *route construction*, involves the incremental generation of a feasible solution to the given problem by means of a route partitioning procedure. The route construction starts with $K_{\min}$, given by (4.3), number of empty sets each representing one forklift route. The procedure iteratively assigns shelves to each route while ensuring that the solution remains feasible in the sense that there exists no subtours, all demand is collected, and no forklift surpasses its load capacity V. The route construction is analogous to the method presented in [3], albeit with the integration of item volume considerations instead of item count.

In the second phase, following the completion of the route construction, the solution computed is improved using the local search heuristic which employs a Variable Negiborhood Search (VNS) strategy to improve the constructed solution from the previous phase. In the third phase, the solution is transformed to contain the number of items to pick instead of the volume quantity by means of another heuristic called *route repair*. This is because it is preferable for the forklift operators to know how many items to pick rather than the total volume to pick. Finally, in phase four, the resulting solution is further improved by solving a TSP for each route using a MIP solver, in order to determine the optimal shelf visitation sequence for the items chosen for the forklift at hand. Some new notations are introduced in this chapter, these are listed in Table 5.1. Figure 5.1 shows a simple flowchart of the heuristic approach.

Table 5.1: *Table of notations for the heuristic approach.*

$\boldsymbol{s}$	An initial solution to the problem computed in the construction phase. This is not necessarily a good solution, but is feasible.
$\boldsymbol{s}^{*,\text{vol}}$	The best solution found in the local search phase. This solution takes into account the quantity of volume each forklift will collect at each shelf.
$\boldsymbol{s}^{*,\text{item}}$	The solution above converted to express quantity of items picked at each shelf, by each forklift. This conversion takes place in phase 3, the route repair. This conversion might also change which shelves the forklifts visit.
$\boldsymbol{s}^{*,\text{final}}$	This is computed in the final step of the heuristic approach, phase four. It is the same as the above solution expressed in number of items to collect, but with optimal order of shelf visits.

The complete heuristic approach is summarised in the following four steps:

1: Initialize all sets and parameters and construct an initial solution, $\boldsymbol{s}$, using route partitioning
2: Improve the solution using a local search procedure and compute $\boldsymbol{s}^{*,\text{vol}}$.
3: Convert solution to show number of items instead of volume to pick, return the solution $\boldsymbol{s}^{*,\text{item}}$.
4: Optimize the order of the shelf visits in the current best solution and return the final solution $\boldsymbol{s}^{*,\text{final}}$.

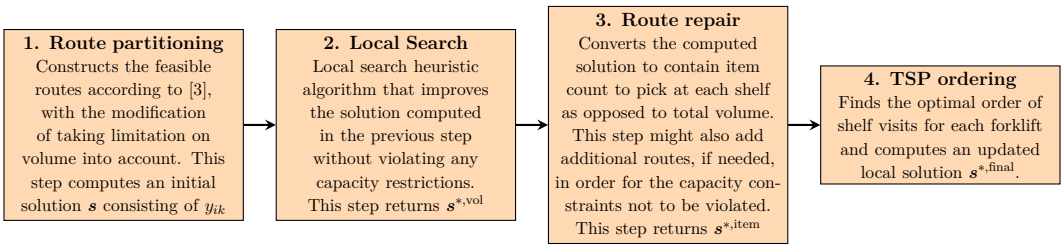


Figure 5.1: *Overview of the four components of the heuristic approach. Phase 3 is completed within phase 2.*

5.1.1 Route partitioning

In this thesis, the construction heuristic proposed in [3], where a set of feasible solutions to the SDVRP are generated, is used as the route partitioning procedure. It is important to emphasise that in [3], the authors assume that the items to be collected are identical in shape, weight, and volume. This, however, is not the case for the problem settings in this thesis but the procedure is analogous.

The route partitioning assigns shelves to forklifts and checks that all routes are feasible. A shelf that is yet to be picked is selected and assigned to one forklift without exceeding V. If the forklift is not loaded upto its maximum capacity, then a new item is added and the process repeats until the forklift is loaded up to its maximum capacity V and all shelf-demands are satisfied. Once the above is completed, the corresponding values of the decision variables $y_{ik},\ i \in \mathcal{N},\ k \in \mathcal{K}$ are

computed, which forms an initial solution $\boldsymbol{s}$ to the routing problem. Following the method described in [3], the first step within the route partitioning initiates the route by selecting the farthest shelf from the depot to begin the route. Next step calculates the capacity threshold and the third step determines how many shelves to evaluate for each iteration. There are several different rules for how each step is to be performed. In [3], nine rules were investigated for step one, three rules for step two and three rules for step three. Thus, in total of 72 possible combinations of rules were tested in [3]. Table 5.2 shows the best best combinations concluded by the authors in [3] and this thesis will use combination number three.

Table 5.2: *Compilation of the nine best combinations out of 72, tested in [3].*

Combination #	Step 1	Step 2	Step 3
1	Farthest shelf	Round-up	Closest shelf
2	Smallest demand	Round-up	Closest shelf
3	Farthest shelf	Round-up	Closest two shelves
4	Farthest shelf	Round-down	Closest two shelves
5	Largest demand	Round-up	Closest two shelves
6	Clarke-Wright	Round-up	Closest two shelves
7	Clarke-Wright	All	Closest two shelves
8	Farthest shelf	Round-up	Closest three shelves
9	Clarke-Wright	Round-up	Closest three shelves

5.1.2 Local search : BVND

The next step for improving the starting solution is carried out using a VNS (Variable Neighborhood Search) strategy. A neighborhood search procedure is a local search method that explores the space of solutions around a current solution and looks for a better one. There are many different types of neighborhood search procedures that can be used, such as 2-opt, 2-opt*, 3-opt, k-opt, λ-interchange, k-transfer and b-cyclic, which involve swapping, inserting, or deleting certain shelves within and between routes in the current feasible solution. In this thesis the 2-opt* method for swapping subtours between routes is used for searching for local minima.

VNS is a simple and effective optimization metaheuristic that has been widely used to solve a variety of combinatorial optimization problems [18, 19]. It is particularly useful for problems where the search space is large and there are many local optima, as it allows the algorithm to escape from local optima and explore different parts of the search space. The key idea behind VNS is to start with an initial solution and perform systematic changes of neighborhoods of that solution during the search space exploration. It will move to a new solution if and only if doing so will result in an improvement of the solution. This process of exploring neighborhoods for better solutions will continue until a local optimum is verified. There are a large variety of strategies when performing a VNS, however in this thesis we will focus on the deterministic variant of the VNS, namely, Basic Variable Neighborhood Descent (BVND).

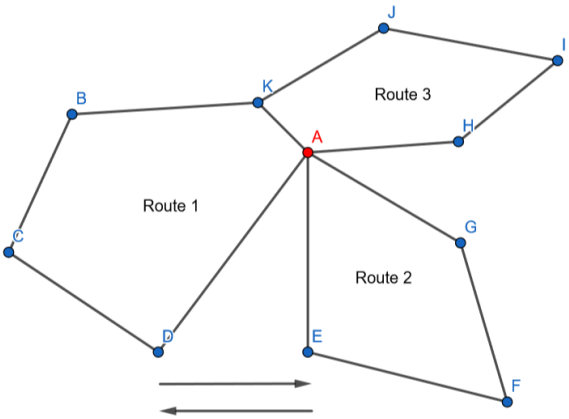


Figure 5.2: *Before swapping D and E*

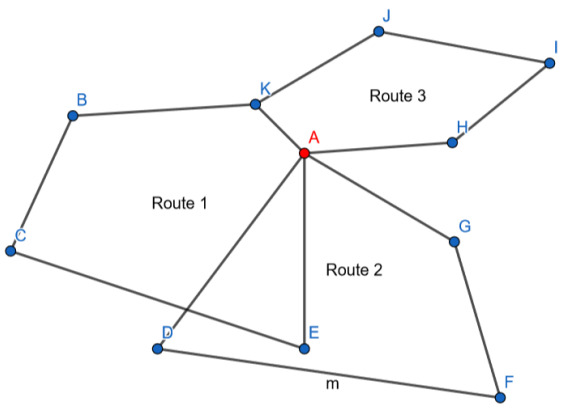


Figure 5.3: *After swapping D and E*

The BVND algorithm takes in the initial solution $\boldsymbol{s}$ and the matrix of distances $\boldsymbol{C}$ between the shelves, the forklift capacity V, and returns the best neighbor solution $\boldsymbol{s}^{*,\text{vol}}$ and its cost. The best neighbor solution is a local minimum and it is the solution yielding the lowest cost. The 2-opt* method iteratively improves a solution by making small changes to it and accepting the change if it results in an improvement. The BVND algorithm repeatedly searches for the best neighbor solution. Algorithm 1 outlines the pseudo code for the local search.

The 2-opt* strategy is simple. When choosing two shelves from two different routes, a swap between them is made and if this results in a lower cost, the swap is kept else the swap is undone. Figures 5.2 and 5.3 shows how such a swap might look like on a network graph.

The following detailed steps are carried out in the local search phase.

1 Initialize the best solution and cost as the current solution and its cost.

2 Repeat steps 2.1–2.7 until no further improvements can be found.

- **2.1** Initialize a flag to indicate whether the solution has improved.
- **2.2** Iterate over all pairs of routes in the current solution.
- **2.3** For each pair of routes, iterate over all pairs of shelves in the routes.
- **2.4** If swapping does not lead to infeasible solution, perform the swap using 2-opt*.
- **2.5** Compute the cost of the modified solution.
- **2.6** If the swap results in a lower cost, set the flag to indicate that the solution has improved, update the best solution and cost, and continue to the next iteration in 2.1.
- **2.7** Otherwise, undo the swap and continue to the next iteration in 2.1.

3 If the flag indicates that the solution has improved, go back to step 2 in the BVDN algorithm.

4 If the flag indicates that the solution has not improved, terminate the search and return the improved solution and the current best cost.

Algorithm 1: 2-opt* neighborhood search procedure (BVND)

Input: $V, \boldsymbol{C}, \boldsymbol{s}$
Output: $\boldsymbol{s}^{*,\text{item}}, f(\boldsymbol{s}^{*,\text{item}})$
Initiate best current solution and best current cost
$\boldsymbol{s}^{*,\text{vol}} := \boldsymbol{s}$
$f(\boldsymbol{s}^{*,\text{vol}}) := f(\boldsymbol{s}^{*})$
while *possible improvements exists* **do**
  foreach *pair of routes in the current solution* **do**
    foreach *pair of shelves* **do**
      Swap the elements between the routes using 2-opt* and call the temporary solution $\boldsymbol{s}^{'\text{,vol}}$.
      Next, calculate $f(\boldsymbol{s}^{'\text{,vol}})$ of the temporary solution, make sure that $\boldsymbol{s}^{'\text{,vol}}$ contains no capacity violations by checking that the sum of each column in $\boldsymbol{s}^{'\text{,vol}} < V$
      if $f(\boldsymbol{s}^{'vol}) < f(\boldsymbol{s}^{*,vol})$ **then**
        $\boldsymbol{s}^{*,\text{vol}} := \boldsymbol{s}^{'\text{,vol}}$
      else
        undo the swap
  if *no further improvement is found* **then**
    break;
return $\boldsymbol{s}^{*,\text{vol}}, f(\boldsymbol{s}^{*,\text{vol}})$

The local search poses a problem since capacity restrictions are decided using volume but we still need the final output to be in number of items picked at the respective shelves. Instructing a forklift operator to pick 40 dm^3 of a certain type of chair is not reasonable. To illustrate this problem with an example, assume there are three forklifts, all with capacities $V = 1000$ dm^3 and eight shelves to be visited. The volumes of items one through eight are $\boldsymbol{v} = (v_1 \ ... \ v_8)^\top = (93 \ \ 58 \ \ 36 \ \ 22 \ \ 72 \ \ 13 \ \ 59 \ \ 65)^\top$ dm^3. An example of the y_{ik} variables in a feasible solution $\boldsymbol{s}^{*,\text{vol}}$ returned by the local search in phase two would be

$$\boldsymbol{s}^{*,\text{vol}}(y) = \begin{array}{l} \\ \text{Shelf 1} \\ \text{Shelf 2} \\ \text{Shelf 3} \\ \text{Shelf 4} \\ \text{Shelf 5} \\ \text{Shelf 6} \\ \text{Shelf 7} \\ \text{Shelf 8} \\ \textbf{Volume Load :} \end{array} \begin{array}{c} \begin{array}{ccc} \text{FL 1} & \text{FL 2} & \text{FL 3} \end{array} \\ \begin{pmatrix} 0 & 266 & 13 \\ 580 & 0 & 0 \\ 72 & 0 & 0 \\ 0 & 110 & 0 \\ 144 & 0 & 0 \\ 0 & 104 & 0 \\ 0 & 0 & 472 \\ 0 & 520 & 0 \end{pmatrix} \\ \begin{array}{ccc} 796 & 1000 & 485 \end{array} \end{array} \tag{5.1}$$

The only demand split is for shelf 1 where forklift 2 picks $y_{12} = 266$ dm^3 and forklift 3 picks $y_{13} = 13$ dm^3. However, the volume of the item at shelf 1 is $v_1 = 93$ dm^3 thus fork lift 2 must pick $266/93 \approx 2.86$ items and fork lift 3 must pick $13/93 \approx 0.14$ items, which is impossible since we are not allowed to split single items. To account for this, the local search heuristic has to be modified with an additional phase, the

route repair phase that converts $\boldsymbol{s}^{*,\text{vol}}$ into a solution $\boldsymbol{s}^{*,\text{item}}$ containing y_{ik}, which indicates the number of items each fork lift should collect at a particular shelf, as opposed to the total volume quantity.

5.1.3 Route Repair

For the the local search to compute a y_{ik} that gives the number of items to be picked, we need to modify $\boldsymbol{s}^{*,\text{vol}}$ using the volumes of the respective items. Let $\oslash$ denote the element-wise division operator between matrices and $\boldsymbol{v} := (v_1\ v_2\ ...\ v_i)^\top,\ i \in \mathcal{N}$ is the ordered vector of volumes for all the items in the picklist. Construct a $N \times K_{\min}$ volume matrix $\boldsymbol{V}$ by placing $K_{\min}$ number of $\boldsymbol{v}$-columns in it: $\boldsymbol{V} := (\boldsymbol{v}\ \ \boldsymbol{v}\ \ ...\ \ \boldsymbol{v})$, the volume-to-item transformation can then be generalized as $\boldsymbol{s}^{*,\text{item}} := \boldsymbol{s}^{*,\text{vol}} \oslash \boldsymbol{V}$. To see this transformation in action, take (5.1) as an example output of $\boldsymbol{s}^{*,\text{vol}}$. In this example, we have that

$$\boldsymbol{V} = \begin{pmatrix} 93 & 93 & 93 \\ 58 & 58 & 58 \\ 36 & 36 & 36 \\ 22 & 22 & 22 \\ 72 & 72 & 72 \\ 13 & 13 & 13 \\ 59 & 59 & 59 \\ 65 & 65 & 65 \end{pmatrix} \tag{5.2}$$

and dividing $\boldsymbol{s}^{*,\text{vol}}$ with $\boldsymbol{V}$ we have the following transformation

$$\boldsymbol{s}^{*,\text{vol}}(y) = \begin{array}{l} \\ \text{Shelf 1} \\ \text{Shelf 2} \\ \text{Shelf 3} \\ \text{Shelf 4} \\ \text{Shelf 5} \\ \text{Shelf 6} \\ \text{Shelf 7} \\ \text{Shelf 8} \\ \textbf{Volume Load}: \end{array} \begin{array}{c} \text{FL 1} \\ \begin{pmatrix} 0 \\ 580 \\ 72 \\ 0 \\ 144 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ 796 \end{array} \begin{array}{c} \text{FL 2} \\ \begin{pmatrix} 266 \\ 0 \\ 0 \\ 110 \\ 0 \\ 104 \\ 0 \\ 520 \end{pmatrix} \\ 1000 \end{array} \begin{array}{c} \text{FL 3} \\ \begin{pmatrix} 13 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 472 \\ 0 \end{pmatrix} \\ 485 \end{array} \implies \boldsymbol{s}^{*,\text{item}}(y) = \begin{array}{l} \\ \text{Shelf 1} \\ \text{Shelf 2} \\ \text{Shelf 3} \\ \text{Shelf 4} \\ \text{Shelf 5} \\ \text{Shelf 6} \\ \text{Shelf 7} \\ \text{Shelf 8} \\ \textbf{Item Load}: \end{array} \begin{array}{c} \text{FL 1} \\ \begin{pmatrix} 0 \\ 10 \\ 2 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ 14 \end{array} \begin{array}{c} \text{FL 2} \\ \begin{pmatrix} 2.86 \\ 0 \\ 0 \\ 5 \\ 0 \\ 8 \\ 0 \\ 8 \end{pmatrix} \\ 23.86 \end{array} \begin{array}{c} \text{FL 3} \\ \begin{pmatrix} 0.14 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8 \\ 0 \end{pmatrix} \\ 8.14 \end{array}. \tag{5.3}$$

The total demand at each shelf is the sum of each row and the total forklift load is the sum of each column. Each picker only visits shelves where there is a non-zero number. We observe that only the first shelf has its volume-demand split across two forklifts and as expected these forklifts are assumed to pick fractional items, as evident by $\boldsymbol{s}^{*,\text{item}}(y)$ in (5.3). To amend this issue, we propose the following method. Define $\langle \xi \rangle := \xi - \lfloor \xi \rfloor$ to be the fractional part of the number $\xi \in \mathbb{R}$. One key insight is that for each row in $\boldsymbol{s}^{*,\text{item}}(y)$, the sum of all fractional parts is always a positive integer. This fact allows us, for each row containing fractional parts, to add the fractional parts and explore how we can further distribute the resulting items across existing forklifts or if more forklifts need to be added. In example (5.3) it is observed that in $\boldsymbol{s}^{*,\text{item}}(y)$ only the first row contains fractional

elements thus truncating their fractional parts gives $\langle \boldsymbol{s}^{*,\text{item}}(y_{12})\rangle = \langle 2.86\rangle = 0.86$ and $\langle \boldsymbol{s}^{*,\text{item}}(y_{13})\rangle = \langle 0.14\rangle = 0.14$. The sum of these fractional parts is $\langle \boldsymbol{s}^{*,\text{item}}(y_{12})\rangle + \langle \boldsymbol{s}^{*,\text{item}}(y_{13})\rangle = 0.86 + 0.14 = 1$, therefore only one item in Shelf 1 needs to be moved among forklifts without causing capacity violations. We now have four different options for proceeding:

1 move $\langle \boldsymbol{s}^{*,\text{item}}(y_{13})\rangle = 0.14$ over to $\boldsymbol{s}^{*,\text{item}}(y_{12}) = 2.86$ and check capacity constraints
2 move $\langle \boldsymbol{s}^{*,\text{item}}(y_{12})\rangle = 0.86$ over to $\boldsymbol{s}^{*,\text{item}}(y_{13}) = 0.14$ and check capacity constraints
3 add $\langle \boldsymbol{s}^{*,\text{item}}(y_{12})\rangle = 0.86$ and $\langle \boldsymbol{s}^{*,\text{item}}(y_{13})\rangle = 0.14$ and place the sum in $\boldsymbol{s}^{*,\text{item}}(y_{11})$ and check capacity constraints
4 create a fourth forklift route and place the sum in $\boldsymbol{s}^{*,\text{item}}(y_{14})$

The fourth option is the last resort and will only be used if none of the current forklifts can take the superfluous item without exceeding V. Clearly, the 0.14 can't be added to 2.86 since the volume capacity of forklift 2 is then exceeded, given that it is already carrying maximum volume capacity. We now have options 2,3,4 left and we note that all the remaining options are a potential workaround. In this particular example, either option 2 or 3 will be used, depending on which choice entails minimal total travel time. Assume that option 3 entails minimal total travel time, the matrix $\boldsymbol{s}^{*,\text{item}}(y)$ in equation (5.3) is updated to only consist of integers as shown in equation (5.4). In order to check that all routes are still within volume capacity, we can apply the inverse of the volume-to-item transform (5.3) on the updated $\boldsymbol{s}^{*,\text{item}}$ as a check to verify that there arecapacity violations:

$$\boldsymbol{s}^{*,\text{item}}(y) = \begin{array}{l} \\ \text{Shelf 1} \\ \text{Shelf 2} \\ \text{Shelf 3} \\ \text{Shelf 4} \\ \text{Shelf 5} \\ \text{Shelf 6} \\ \text{Shelf 7} \\ \text{Shelf 8} \\ \textbf{Item Load :} \end{array} \begin{array}{c} \text{FL 1} \\ \left(\begin{array}{c} 1 \\ 10 \\ 2 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{array}\right. \\ 15 \end{array} \begin{array}{c} \text{FL 2} \\ \begin{array}{c} 2 \\ 0 \\ 0 \\ 5 \\ 0 \\ 8 \\ 0 \\ 8 \end{array} \\ 23 \end{array} \begin{array}{c} \text{FL 3} \\ \left.\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 8 \\ 0 \end{array}\right) \\ 8 \end{array} \implies \boldsymbol{s}^{*,\text{vol}}(y) = \begin{array}{l} \\ \text{Shelf 1} \\ \text{Shelf 2} \\ \text{Shelf 3} \\ \text{Shelf 4} \\ \text{Shelf 5} \\ \text{Shelf 6} \\ \text{Shelf 7} \\ \text{Shelf 8} \\ \textbf{Volume Load :} \end{array} \begin{array}{c} \text{FL 1} \\ \left(\begin{array}{c} 93 \\ 580 \\ 72 \\ 0 \\ 144 \\ 0 \\ 0 \\ 0 \end{array}\right. \\ 889 \end{array} \begin{array}{c} \text{FL 2} \\ \begin{array}{c} 186 \\ 0 \\ 0 \\ 110 \\ 0 \\ 104 \\ 0 \\ 520 \end{array} \\ 920 \end{array} \begin{array}{c} \text{FL 3} \\ \left.\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 472 \\ 0 \end{array}\right) \\ 472 \end{array}. \tag{5.4}$$

In the updated $\boldsymbol{s}^{*,\text{vol}}$ after applying the inverse, in equation (5.4), one observes that all forklifts are within maximum volume capacity and in $\boldsymbol{s}^{*,\text{item}}$ all elements are now integers. The following steps are used in the route repair and Algorithm 2 outlines the pseudo code.

1 Perform the element wise division $\boldsymbol{s}^{*,\text{vol}} \oslash \boldsymbol{V}$ to transform the current solution in terms of volume to collect, to a new solution $\boldsymbol{s}^{*,\text{item}}$ in terms of number of items to collect.
2 Identify all elements in $\boldsymbol{s}^{*,\text{item}}$ that are not integers and truncate the elements to integers, then add all the decimals. Due to the nature of the route construction algorithm, the decimals in each row will always sum to integers. Distribute this

integer demand over forklift routes yielding the least cost while not exceeding volume capacities.

3 If no further integers can be added to any of the existing forklift routes without exceeding V, a new forklift route must be added and the superfluous integers are placed in the new route. This is an iterative process and depending on the problem instance, many new routes can potentially be added. This changes the dimensions of $\boldsymbol{s}^{*,\text{item}}$ to be $N \times (K_{\min} + a)$, where $a \in \mathbb{N}_+$ is the number of additional routes added in the iterative process. It is assumed that Ellos always have more forklifts than $K_{\min}$ to their disposal.

4 Once all routes are within volume capacity and $\boldsymbol{s}^{*,\text{item}}$ only contains integers and the locally optimal solution in terms of number of whole items $\boldsymbol{s}^{*,\text{item}}$ is returned.

Below is the pseudo code for the route repair procedure, Algorithm 2.

Algorithm 2: Route Repair

Input: $\boldsymbol{C}$, $\boldsymbol{s}^{*,\text{vol}}$, $\boldsymbol{v}$
Output: $\boldsymbol{s}^{*,\text{item}}$, $f(\boldsymbol{s}^{*,\text{item}})$
$\boldsymbol{V} = (\boldsymbol{v} \;\; \boldsymbol{v} \;\; ... \;\; \boldsymbol{v})^\top$
$\boldsymbol{s}^{*,\text{item}} = \boldsymbol{s}^{*,\text{vol}} \oslash \boldsymbol{V}$
foreach $k \in \mathcal{K}$ **do**
    foreach $i \in \mathcal{N}$ **do**
        Sum all the $\langle \boldsymbol{s}^{*,\text{item}}(y_{ik}) \rangle > 0$ and denote the sum $\Sigma \in \mathbb{N}_+$.
        Distribute the integer Σ across routes with residual capacity
        such that the cost is minimal
    if Σ *can't be distributed fully across all existing routes* **then**
        Add new routes iteratively until all demands satisfied and no route exceeds V
return $\boldsymbol{s}^{*,\text{item}}, f(\boldsymbol{s}^{*,\text{item}})$ where the y component only contains integers.

The final phase is to further improve the solution $\boldsymbol{s}^{*,\text{item}}$ by calculating the optimal order of shelf-visitation for each forklift using a TSP solution procedure. This phase can be solved using many efficient TSP solvers that are able to solve large TSP instances with good solutions or in many cases to optimality very fast, one example is the GENIUS-CP algorithm [20]. Here, Gurobipy is used. The details of solving a TSP is well known in the literature and easy to implement; we will therefore omit the details here. This phase computes and returns the final solution $\boldsymbol{s}^{*,\text{final}}$. This concludes the heuristic approach.

6
Tests and results

6.1 Implementation and test settings

The model presented in Chapter 4 and the heuristic developed in Chapter 5 were implemented in Python using the Gurobi Python interface `gurobipy` that uses the Gurobi solver in a 64-bit environment. The computer used is powered by a Quad Core Intel i7 1.90 GHz processor with eight threads and 16 GB RAM. The maximum runtime is 180 minutes. The SDVRP model was first run for 30 mins and then for 180 mins in order to see how the lower and/or upper bounds improve. The different problem instances are summarized in Table 6.1

Table 6.1: *Problem instances and descriptions.*

Instance	# Shelves	# Units	# Forklifts	Avg vol/unit (dm^3)	Forklift capacity (dm^3)
A	10	57	3	48.9	1500
B	20	110	3	50.6	2000
C	50	258	7	55.7	2500
D	100	572	10	51.5	3000
E	250	1420	21	47.0	3500
F	500	2760	39	50.3	4000

6.1.1 Test results

Table 6.2 shows the results when solving the mathematical model using the gurobi solver with time limits 30 mins and 180 mins respectively. Increasing the run time from 30 mins to 180 mins did not improve the bounds by much.

Table 6.3 shows the results when running the heuristic on the same problem instances. For problem instances A and B, the heuristic gives a higher objective value than the Gurobi solver. For problem instances C-F, the heuristic objective indicate better results while maintaining a very low run time in all instances. The improvement achieved by employing the heuristic as compared to the exact solution of the mathematical model is more significant as the size of the instances increase.

Table 6.4 shows how the objective value is sequentially improved during each step of the heuristic approach. We observe that the improvement (cost savings) are greater the larger the problem instance. The last column to the right indicates the total cost savings from the construction objective to the TSP objective.

As explained earlier in the thesis, the heuristic approach might, depending on the combination of demands and volumes, add more routes to our solution in order

to guarantee that capacity constraints are not violated. It can also be interesting to see how many new routes the heuristic approach actually adds for each instance. To do this, simulations were used. For each instance 1000 different sets of demands and volumes were randomly generated from normal distributions $N(\mu_{\text{units}}, 1), N(\mu_{\text{vol}}, 1)$ with the mean chosen from Table 6.1, #Units and Avg vol/unit, for each instance. The heuristic never added more than one route in any simulation. Figure 6.3 shows a bar plot of the results.

Figure 6.2 shows a comparison between Ellos current picking policy, the S-shape heuristic, against the heuristic approach presented in this thesis. The cost savings are evident particularly for large problem instances.

Table 6.2: *Results from using the SDVRP model using the Gurobi solver.*

Problem instance	GAP	Objective value	Lower bound	Runtime (min)
A	0.00 %	851.52	851.52	9
B	30.46 %	1245.13	865.87	30
C	50.77 %	3711.41	1827.30	30
D	68.63 %	7174.43	2250.46	30
E	93.33 %	73374.58	4892.55	30
F	98.12 %	738891.65	13869.31	30
A	0.00 %	851.52	851.52	9
B	25.88 %	1202.31	891.19	180
C	42.89 %	3344.82	1910.43	180
D	54.47 %	5259.51	2394.81	180
E	92.57 %	69547.21	5164.72	180
F	97.93 %	725714.11	15004.56	180

Table 6.3: *Results for the heuristic approach. The last column indicates the change in the objective value when comparing the objective value given by the Gurobi solver (180 minute run) in Table 6.2, to the objective value given by the heuristic approach. Negative values here correspond to lower heuristic objective value compared to the objective value given by the Gurobi solver.*

Problem instance	Heuristic objective	Runtime (min)	Relative change
A	946.07	< 1	-9.99 %
B	1374.32	< 1	-12.52 %
C	3181.94	< 1	5.12 %
D	5101.03	< 1	3.11 %
E	10031.54	< 1	593.3 %
F	16947.44	< 2	4182.63 %

Table 6.4: *Comparison of objective values at different steps in the heuristic. The Reduction-column is the reduction in objective value from the first route partition phase to the last TSP phase.*

Problem instance	Phase 1 Objective	Phase 2-3 Objective	Phase 4 Objective	Reduction
A	1283.55	944.11	946.07	26.29 %
B	3378.09	2122.41	1374.32	37.17 %
C	6141.00	4231.84	3181.94	48.19 %
D	11683.39	6389.63	5101.03	56.34 %
E	25415.77	12681.32	10031.54	60.53 %
F	45131.17	22302.02	16947.44	62.45 %

Figure 6.1: *Bar plot of objective values at different steps in the heuristic approach including the model objective obtained from the Gurobi solver for the 180 min run. Observe that the model objective for problem instance F has been capped. The TSP objective is equivalent to the heuristic objective.*

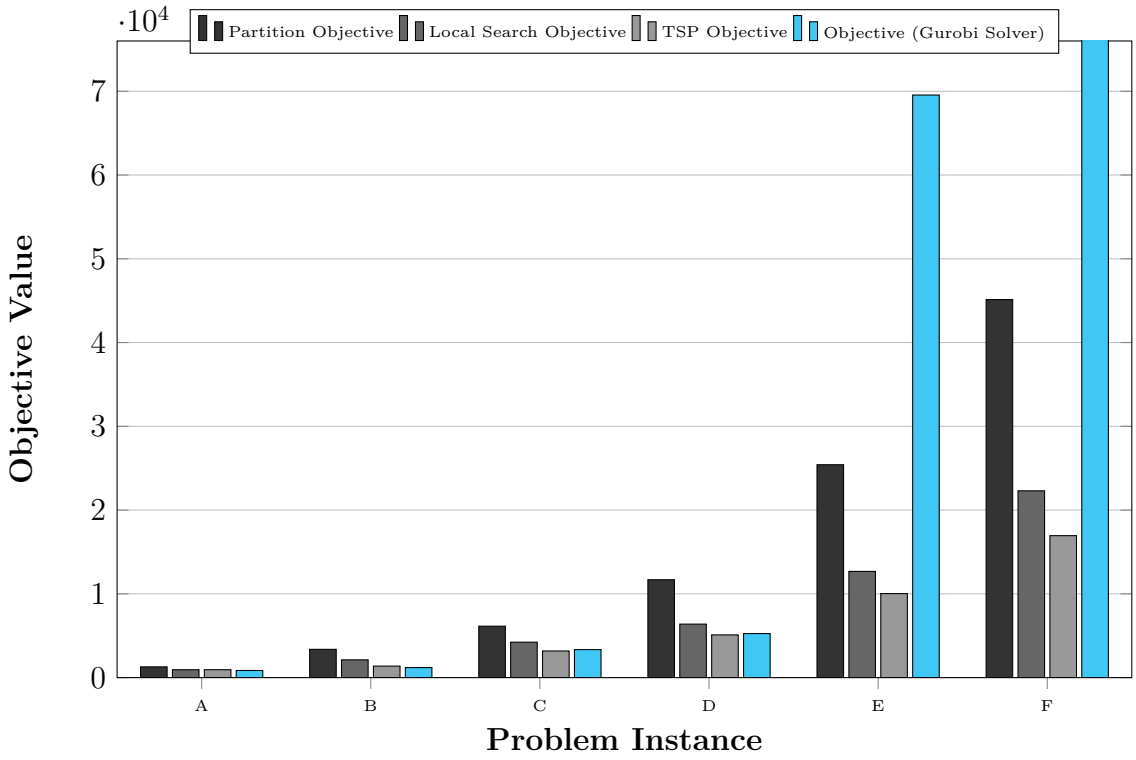


Figure 6.2: *Bar plot of objective values for the S-shape picking policy currently used at Ellos Group compared to the Heuristic approach.*

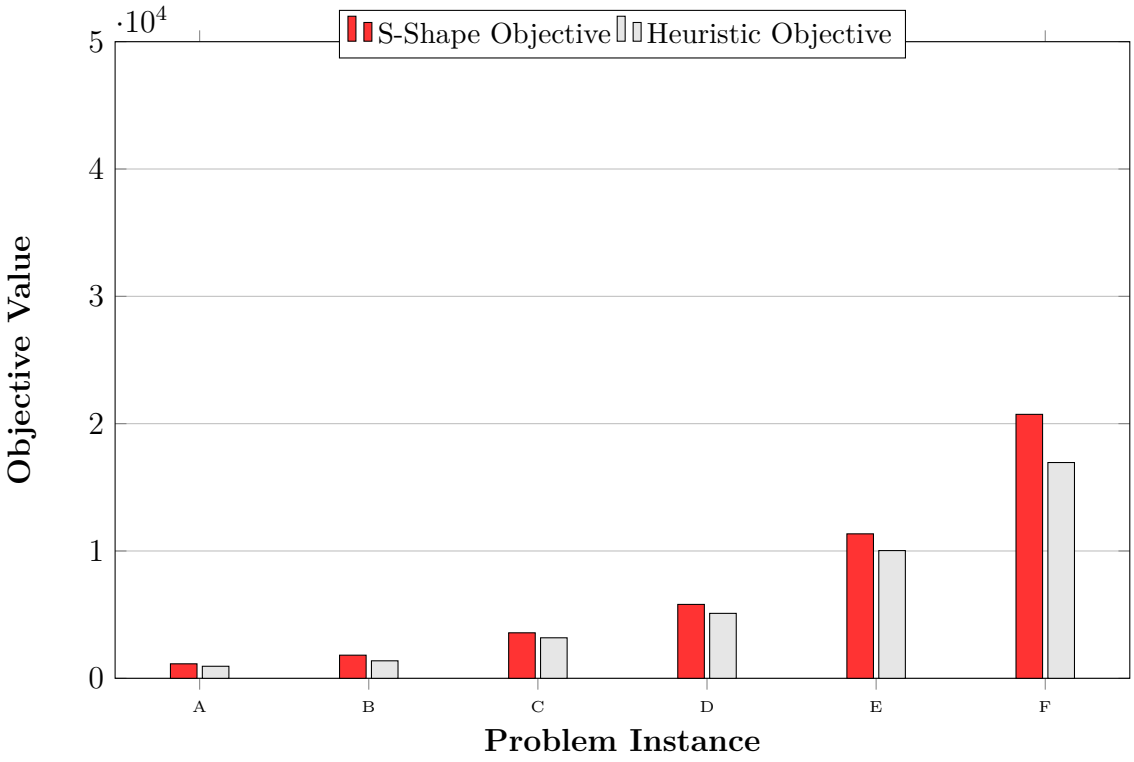
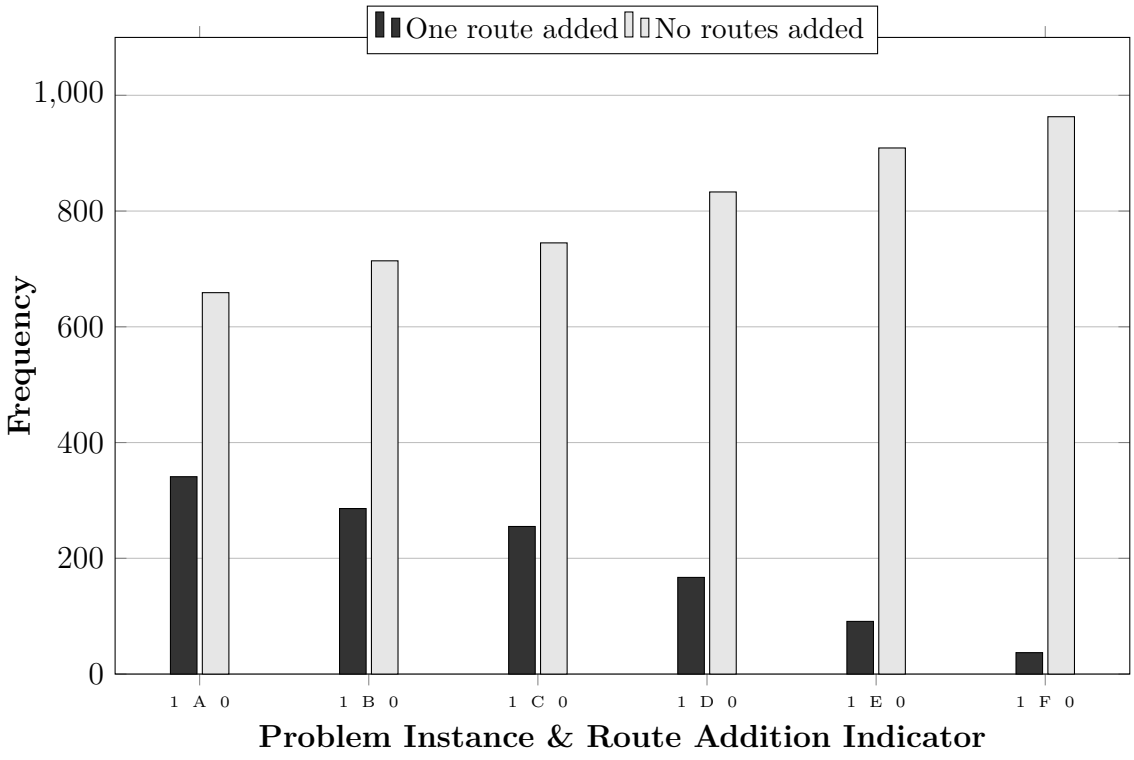


Figure 6.3: *Bar plot indicating how many times one extra route was added out of 1000 simulations.*



7
Discussion

The local search method in this thesis uses the deterministic variable neighbor strategy, which means that it does not use randomness to explore the search space. Instead, it relies on a greedy approach, where at each step it selects the neighbor solution with the lowest cost and moves to it.

While this approach can be effective in finding local minima, it can also get stuck in a local minima if the current solution is already a local minima and there are no better neighbors to move to. In such cases, the algorithm will not be able to find a better solution and will terminate immediately. Note that this will not necessarily make the heuristic approach bad in any way but would rather indicate that the current solution is already fairly good.

One way to escape local minima is to use a variant of VNS called Simulated Annealing, which introduces a random element to the search process by allowing the algorithm to sometimes accept solutions with higher costs. This helps the algorithm to explore the search space more widely and potentially find better solutions that are not immediately accessible through the local neighborhood.

The performance of the proposed construction heuristic is evaluated against the mathematical representation and the current picking policy at Ellos Group. The results of the evaluation show that the heuristic produces solutions in significantly less computation time for all problem instances when compared to the mathematical implementation, despite it producing worse results for problem instances A and B. For the smaller instances A, B, and C the cost difference between the heuristic and Ellos current picking policy is not so large, however it increases with problem size. The cost savings, when compared tot he S-shaped heuristic, were on average between 8%–11%, depending on the size of the problem instance.

In addition to being used as a stand-alone solution approach, the heuristic can be used as a starting point for further optimization using other methods, such as tabu search, column generation or the two-step method. These methods can then be applied to improve the initial solution provided by the heuristic, potentially resulting in even better solutions. The solution obtained from the heuristic can also be used as an input to a MILP solver such as Gurobi, since the value of a good, feasible solution can be used to cut branches of the branch-and-bound tree early in the process. This will reduce the computation time considerably since there are far less nodes of the tree to be explored.

Moreover, the heuristic does not assume symmetric distances, meaning that it can be applied to problems where the distance between two locations may be different in one direction compared to the other. This makes the heuristic more flexible and applicable to a wider range of problems.

As seen in Figure 6.3, the number of times a new route is added by the heuristic increases with increasing instance sizes. To explain this, we can look at Table 6.1 where we see that the larger the instance the greater the capacity of the forklift, but on the other hand the average volume per unit is roughly constant through all the problem instances. Since the capacities are larger, there are more ways to redistribute the items picked among the forklifts thus less additional forklifts are required.

Bibliography

[1] 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. doi: 10.1016/j.ejor.2006.07.009.

[2] C. Archetti and M. G. Speranza. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19(1–2):3–22, 2012. doi: 10.1111/j.1475-3995.2011.00811.x.

[3] Joseph Hubert Wilck IV and Tom M. Cavalier. A construction heuristic for the split delivery vehicle routing problem. *American Journal of Operations Research*, 02(02):153–162, Mar 2012. doi: 10.4236/ajor.2012.22018.

[4] 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. doi: 10.1016/j.ejor.2006.07.009.

[5] Sunney Fotedar, Torgny Almgren, Stefan Cedergren, Ann-Brith Strömberg, and Michael Patriksson. A Mathematical Optimization of the Tactical Resource Allocation of Machining Resources for an Efficient Capacity Utilization in Aerospace Component Manufacturing. pages 183–188, October 2019. doi: 10.3384/ecp19162021. URL `https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=162&Article_No=21`.

[6] F. A. Ficken. *The simplex method of linear programming*. Dover Publications, Inc, Mineola, New York, dover edition edition, 2015. ISBN 9780486796857.

[7] Michele Conforti, Gerard Cornuejols, and Giacomo Zambelli. *Integer Programming*. Number 271 in Graduate Texts in Mathematics. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2014 edition, 2014. ISBN 9783319110080.

[8] Stephanie. Traveling salesman problem and tsp art, Apr 2023. URL `https://www.statisticshowto.com/traveling-salesman-problem-and-tsp-art/`.

[9] URL `https://www.theorsociety.com/about-or/or-methods/heuristics/a-brief-history-of-the-travelling-salesman-problem/`.

[10] Rajesh Matai, Surya Singh, and Murari Lal. Traveling salesman problem: An overview of applications, formulations, and solution approaches. *Traveling Salesman Problem, Theory and Applications*, page 6, Jul 2010. doi: 10.5772/12909.

[11] Abraham P. Punnen. The traveling salesman problem: Applications, formulations and variations. pages 1–28, 2007. doi: 10.1007/0-306-48213-4_1.

[12] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959. doi: 10.1287/mnsc.6.1.80.

[13] Moshe Dror and Pierre Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989. doi: 10.1287/trsc.23.2.141.

[14] Moshe Dror and Pierre Trudeau. Split delivery routing. *Naval Research Logistics (NRL)*, 37(3):383–402, 1990. doi: 10.1002/nav.3800370304.

[15] C. Archetti, N. Bianchessi, and M. G. Speranza. A column generation approach for the split delivery vehicle routing problem. *Networks*, 58(4):241–254, 2011. doi: 10.1002/net.20467.

[16] Bin Ji, Saiqi Zhou, Samson S. Yu, and Guohua Wu. An enhanced neighborhood search algorithm for solving the split delivery vehicle routing problem with two-dimensional loading constraints. *Computers & Industrial Engineering*, 162: 107720, December 2021. ISSN 03608352. doi: 10.1016/j.cie.2021.107720. URL `https://linkinghub.elsevier.com/retrieve/pii/S0360835221006240`.

[17] Tolga Bektaş and Luis Gouveia. Requiem for the Miller–Tucker–Zemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832, 2014. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2013.07.038. URL `https://www.sciencedirect.com/science/article/pii/S037722171300619X`. Vehicle Routing and Distribution Logistics.

[18] Pierre Hansen, Nenad Mladenović, and José A. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, March 2010. ISSN 0254-5330, 1572-9338. doi: 10.1007/s10479-009-0657-6. URL `http://link.springer.com/10.1007/s10479-009-0657-6`.

[19] Abraham Duarte, Jesús Sánchez-Oro, Nenad Mladenović, and Raca Todosijević. Variable neighborhood descent. *Handbook of Heuristics*, page 341–367, 2018. doi: 10.1007/978-3-319-07124-4_9.

[20] Gendreaul Michel Rousseau Jean-Marc Pesant, Gille. Genius-cp: A generic single-vehicle routing algorithm. In Gert Smolka, editor, *Principles and Practice of Constraint Programming-CP97*, pages 420–434, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69642-1.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY