Models and Algorithms for Inbound and Outbound Truck to Door Scheduling

Sayed Ibrahim Sayed

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Industrial Engineering) at

Concordia University

Montréal, Québec, Canada

June 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sayed Ibrahim Sayed**

Entitled: **Models and Algorithms for Inbound and Outbound Truck to Door Scheduling**

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

Master of Applied Science (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 Committee:

______________________ Chair
Dr. Robin Drew

______________________ External Examiner
Dr. Navneet Vidyarthi, JMSB

______________________ Examiner
Dr. Onur Kuzgunkaya, MIE

______________________ Supervisor
Dr. Ivan Contreras

Approved by ______________________
Martin D. Pugh, Chair
Department of Mechanical and Industrial Engineering

__________ 2019 ______________________
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Models and Algorithms for Inbound and Outbound Truck to Door Scheduling

Sayed Ibrahim Sayed

Cross-docking is a logistic strategy that facilitates rapid movement of consolidated products between suppliers and retailers within a supply chain. It is also a warehousing strategy that aims at reducing or eliminating storage and order picking, two of which are known to be major costly operations of any typical warehouse. This strategy has been used in the retailing, manufacturing, and automotive industries. In a cross-dock, goods are unloaded from incoming trucks, consolidated according to their destinations, and then, loaded into outgoing trucks with little or no storage in between.

In this thesis, we address an integrated cross-dock door assignment and truck scheduling problem in which the assignment and sequencing of incoming trucks to strip doors and outgoing trucks to stack doors is optimized to minimize the total time to process all trucks. We present a mixed integer programming formulation to model this problem and some valid inequalities to strengthen the formulation. We also present two metaheuristics to obtain high quality solutions in reasonable CPU times. These algorithms use a mix of composite dispatching rules, constructive heuristics, local search heuristics which are embedded into a greedy randomized adaptive search procedure (GRASP) and an iterated local search (ILS). Results of computational experiments are presented to assess the performance of the proposed algorithms, in comparison with a general purpose solver.

Acknowledgments

All praise belongs to Allah who has given me the patience and perseverance to accomplish my master's degree. Peace upon his messenger, Muhammad.

I want to express my very great appreciation to my supervisor Dr. Ivan Contreras for his guidance and support to finish my thesis. The advice given by him has been a great help in accomplishing and writing this thesis.

I want to express my gratitude towards my beloved family as nothing would be possible without their support and encouragement. They helped me a lot to reach this stage in my life. My parents supported me emotionally and financially. Thanks to my father Sayed Omar and my mother Fozia Sayed for the continues encouragement, motivation, and supports. Thanks to my sisters Amina, Sawsan, Noorah, and Fatema for motivation and help. Also, thanks to my brother's Sultan, Yousuf, and Abdulrahman for their puerile support.

I want to thank my friends and lab-mates for their help in writing the thesis. I can not forget the valuable support and motivation of Maryam. I have often looked towards her for a valuable suggestion, and she always helped me whenever I needed an assistant. Thanks to my friend Aditya, a PhD candidate for sharing knowledge through the course work and having significant dissections about the research. To my friend since my childhood, Amjad thanks for your support and motivation. Thanks again to Maryam, Amjad and Aditya for fixing the grammatical issues. Thanks to Wael for greet dissections and valuable suggestions about my thesis to Carlos for the great memories in office.

Thanks to Concordia University, Transportation Association of Canada, and port of

Montreal for the financial support during my study. At last, thanks to many other people whose names are not mentioned here but contributed to this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Globalization and competition have been constantly putting pressure on companies and organizations to have efficient supply chain management. PwCs Performance Measurement Group classifies some companies to be Best in Class supply chains (BIC). Companies classified as BIC obtain 20% higher profitability and reach 50% higher sales growth. According to Supply Chain Management Association (2016), in Canada, the size of the supply chain sector is equal to that of the manufacturing sector and around six times the size of the agriculture sector. Supply Chain Management Association (2016) defines the supply chain management (SCM) as strategic management of the relationship between several groups or divisions, the flow of materials, services, information, and money to achieve better economic value. According to Transport Canada (2017), more than 800 millions tons of goods were transported across the country in 2015 by private sectors. Logistic management is defined as the the strategic management of flow of materials from end to end. Good logistic practices result in lower transportation costs, better customer satisfaction, and higher efficiency in terms of speed, flexibility, and capacity Roy (2011). In the business sector, different companies are applying different transportation and logistics strategies. The main four strategies are direct shipment, milk-runs, warehousing, and cross-docking. The commodities are shipped directly from the origin to the destination in a direct shipment strategy.

In a milk-run strategy, commodities are grouped into routes, where each route consists of a sequence of pickups and drop offs. In cross-docking and warehousing strategies, products are consolidated into full truckload (Buijs et al., 2014).

In transportation networks, cross-dock terminals are the intermediate nodes where products are consolidated into full truckload to achieve economies of scale. At the cross-dock terminal, the products are not stored for a long time as they are in traditional warehouses. Elimination of storage results in zero holding cost, minimal damage, and loss of items. In comparison to a point to point delivery system, there is a trade-off between the handling costs at cross-dock terminal and savings from achieving full truckloads. In supply chains where suppliers and/or customers are sparsely located from one another, cross-dock technique results in an efficient transportation system as compared to the point to point delivery system. The benefits to be yielded from the cross-dock technique depends on certain conditions. According to Apte and Viswanathan (2000), products with the following characteristics will best fit the cross-dock technique: products with high demand rates, products with stable rates, and products with low stock-out costs.

To illustrate the process at the cross-dock terminal, an inbound truck arriving at the terminal is assigned to an inbound door, where the commodities are unloaded from the inbound truck at the inbound door in the unloading area. The unloaded commodities are sorted and consolidated at the sorting area and then moved on to the loading area. On the other hand, the outbound truck is assigned to an outbound door. Once assigned, the commodities from the loading area are then loaded on to the outbound truck to be shipped to their next destination. Unloading, loading and moving the commodities within the cross-dock terminal can be done by forklifts or by advanced automated systems. The efficiency of the cross-dock terminal is directly affected by how these trucks are assigned and scheduled. (Stephan and Boysen, 2011).

A successful implementation of the cross-docking technique includes solving many decision problems. Buijs et al. (2014) proposed a general classification of the cross-dock problems. As shown in Figure 1.1, problems are classified into two main classes, local cross-dock management problems and cross-docking network management problems. The first class focuses on the activities within a cross-dock terminal whereas the second one focuses on the activities between the cross-dock terminal and other participating entities interacting with the cross-dock terminal. These are then further classified into following three sub-classes: strategic, tactical and operational. Decisions involved in designing the cross-dock terminal include the shape of the cross-dock, number of doors, capacity and material handling design. The network design decisions include the terminal location, the terminal type, and the number of cross-dock terminals. In the tactical level, cross-dock planning decisions include trucks to doors assignment and re-sources capacity planning. On the other hand, network planning decisions contain the route capacity planning, shipment flow allocation and assignment of shipments to destinations. At the operational level, cross-dock scheduling problem studies the trucks scheduling and resource scheduling. Network scheduling includes solving the truck routing problem. Here, we study a synchronized truck scheduling and truck to door assignment problem.

According to Ladier and Alpan (2016), truck to door scheduling problem in cross-dock is a combination of two decision problems: the cross-dock door assignment problem and the truck scheduling in the cross-dock problem. Cross-dock door assignment problem deals with the assignment of trucks to doors in such a way that minimizes or maximizes the required objective while respecting the constraints in the place. The truck scheduling problem aims to determine the time table at the trucks which are scheduled. Also, it assumes that the assignment of trucks to doors are known in advance. Another version of this problem is the truck sequencing problem, where the decision to be made is the sequencing

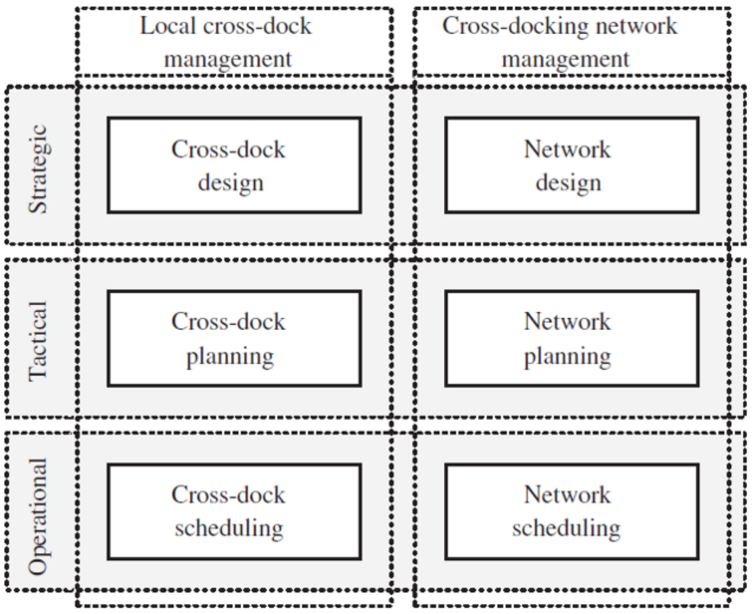

Figure 1.1: Buijs et al. (2014) Cross-docking problem classes.

of trucks irrespective of the time at which trucks are scheduled. The truck to door scheduling problem deals with both the assignment of trucks to doors and the time at the trucks which are scheduled. The objective could be to minimize the cost, time, or the number of delayed trucks. These problems are further classified into three classes, only inbound problem, only outbound problem and both inbound & outbound problem. For instance, in the inbound cross-cock door assignment problem, the decisions to be made are for the inbound side only, and the assignment of the outbound trucks is assumed to be known and vice versa for the outbound cross-dock door assignment problem. On the other hand, for both inbound and outbound Cross-Dock door assignment problem, the assignment of both inbound and outbound trucks should be determined. Both the assignment problems and scheduling problems are well known to be hard problems to solve even in the single side case. (Ladier and Alpan, 2016).

In this thesis, we study the *truck-to-door scheduling problem* (TDSP). In this problem, we are aiming to reach an optimum decision for the assignment and scheduling for a given set of inbound and outbound trucks to a set of inbound and outbound doors. One of the assumptions in this problem is that the distance between the inbound door and the outbound door is a factor to be considered. This is due to the fact that typically, the number of doors in a cross-dock terminal can range from anywhere between 8 to 200 doors. While in some cases, it could be 500 or more doors e.g. Dallas, TX Terminal which has more than 500 doors (Gue, 1999). Incorporating that into the scheduling problem leads to minimizing the handling required to run the operation. Also, we assume that trucks are available at time zero, the truck permutation is not allowed, and products are not interchangeable. Moreover, we consider an exclusive mode which means that each door serves either as an inbound door or outbound door, but not both at same time. According to the classification scheme presented by Boysen and Fliedner (2010), our problem is classified as $[E|t_{i0}|C_{max}]$. The principal contributions of this thesis are:

- we introduce the TDSP problem where we consider certain assumptions to have a closer representation of the reality.
- We present two mathematical programming formulations to for the TDSP.
- we propose two constructive algorithms to obtain feasible solutions.
- we develop a local search algorithm using various neighbourhoods.
- we develop two metaheuristics to obtain high quality solutions.

The remainder of this thesis is organized as follows. In Chapter 2, a comprehensive literature review on Truck-to-door scheduling at cross-dock problems is presented. Chapter 3 contains the problem description, assumptions, notations and two mathematical models. In Chapter 4, we present the solution methodologies and computational results. In the computational results section, we generate instances and analyze the results obtained from the two formulations and proposed solution methodologies. Finally, we draw our conclusion in Chapter 5.

Chapter 2

Literature Review

In this chapter, we present a literature review for truck to door scheduling problems in a cross-dock terminal. Our focus here is on the existing research in scheduling and sequencing where the decisions to be made are for both inbound and outbound trucks at the cross-dock terminal. Boysen and Fliedner (2010) present a classification scheme and notations for structuring this complex research field. The notations are a combination of classical machine scheduling attributes and additional cross-dock scheduling attributes. The classification scheme and notations are based on both door environment and operational characteristics of the cross-dock. Also, they present a literature review for cross-dock problems. Stephan and Boysen (2011) present the cross-dock concept and classified decision problems and for each class, they provide a detailed literature review.

Van Belle et al. (2012) present a classification of problem types for cross-dock research by conducting an extensive review of the literature. Buijs et al. (2014) provide a framework for synchronizing interdependent cross-dock problems to support the future work. They also present a literature review and classification based on the inputs and outputs of the decision problem of cross-dock research. They categorized decision problems at the cross-dock into two main types: local decisions and network management decisions. Furthermore, for each type, they classified decision problems into three decision level:

strategic, tactical, or operational.

Ladier and Alpan (2016) study the gap between the research and the industry by analyzing and matching publications and literature reviews to industrial practices. They focus on tactical and operational decisions at the cross-dock terminal. They visited eight cross-dock terminals and interviewed managers there. For two out of eight managers, makespan is an important performance measurement. They conclude that, in practice, its preferable to separate inbound and outbound doors. They highlighted the gap between the research and the industry mentioned above. To fill the gap between researches and the industry practices, they suggested some areas to focus.

We overview the work done on the single door environment where authors assumed that the terminal consists of one inbound door and one outbound door. After that, we overview works done on the multiple door environments under the assumption that the internal transpiration time is neglected. Following that, an overview of the work done on the multiple door environment is given, where the internal transportation time is taken into account.

2.1 Single Docks at Each Side

To the best of our knowledge, Yu (2002) is the first work to study the truck scheduling problem. It is a PhD thesis and it studies three different versions of the problem. The first problem, that is also presented in Yu and Egbelu (2008), assumes that the storage is allowed and the truck preemption is not allowed. The second problem assumes that the storage is not allowed while the truck preemption is allowed. The last problem assumes that both storage and truck preemption are allowed. All problems assume that trucks are available at time zero, trucks change over time and the product shifting time is identified and fixed, and products are interchangeable. The thesis presents mixed integer programming (MIP) formulations and heuristics to solve the problem. For the first problem, Vahdani and Zandieh

(2010) and Arabani et al. (2011) propose metaheuristics algorithms. Also, Shiguemoto et al. (2014) propose a hybrid genetic algorithm with the tabu search. Keshtzari et al. (2016) propose a new MIP formulation model. They compare their formulation with the one proposed by Yu and Egbelu (2008) using the commercial software CPLEX that was able to solve their model in significantly less amount of time. Also, they propose a meta-heuristic to solve larger instances.

Chen and Lee (2009) and Chen et al. (2009) study the cross-dock sequencing problem with a single inbound and single outbound door. The problem is studied in the context of the classical two-machine flow shop with additional precedence constraints. They assumed that trucks are available at time zero, products are not interchangeable, and the objective is to minimize the makespan. They introduce some optimality properties and show that the problem is NP-hard, however in some particular cases the problem can be solved in polynomial time. They propose both heuristics and branch and bound algorithms. Fonseca et al. (2017) present a time-index formulation and propose a Lagrangian relaxation that decomposes the problem into two sub-problems, one for each side. To solve the sub-problems, they present a polynomial time algorithm. Moreover, they present hybrid Lagrangian meta-heuristics. Then, they extended that by studying multiple inbound and outbound doors.

Boysen et al. (2010) present a time-index formulation and assume that loading and unloading of a truck takes one time slot. Also, they assume that all trucks takes same process time and neglect the transportation time at the terminal. They prove that the problem is NP-hard and propose a method to obtain a lower bound and a decomposition approach to get good upper bounds. They decompose the problem into two sub-problems: one for the inbound and the other for the outbound side. Bodnar et al. (2015) study a scheduling problem with the time windows constraint. They consider a cross-dock terminal with three types of docks, inbound docks, outbound docks and mix docks. They assume that products unloaded from an inbound are either loaded directly to the outbound truck or stored at the temporary storage area. They presented a MIP formulation and heuristics.

2.2 Multiple Docks Without Internal Transshipment Time

Song and Chen (2007) study scheduling parallel inbound doors and single outbound doors, in a flow shop context. In their model, preemption is not allowed, all trucks are available at time zero, and products are not interchangeable. They present a MIP formulation, two heuristics based on Johnson's rule and two methods to obtain a lower bound.

Chen and Song (2009) consider a parallel dock door on at least one side, which is a generalization of the problem studied in Chen and Lee (2009). They study the problem in the context of the two-stage hybrid flow shop problem. They neglected the transportation and consolidation time at the terminal. They proposed a MIP model, developed four heuristics based on Johnsons rule and provided a method to get a lower bound. Cota et al. (2016) study the same problem and present a time-index formulation that dominates the one proposed by Chen and Song (2009). Also, they present a constructive algorithm that outperforms the heuristics presented in Chen and Song (2009) for large instances. Bellanger et al. (2013) study the problem as a three-stage hybrid flow shop. They consider a cross-dock terminal with parallel inbound and outbound doors and they include the sorting process by considering a parallel station between inbound and outbound doors. They present several heuristics and a branch and bound algorithm to solve the problem.

Li et al. (2004) study the truck sequencing problem in a just in time context. They assume that products are not interchangeable, the preemption is not allowed and both release date and due dates are known. The objective is to minimize the total weighted earliness and the tardiness of the trucks. They present a MIP formulation and provide a heuristic to solve the problem. Joo and Kim (2013) consider three types of trucks; inbound trucks, outbound trucks and a mix of both trucks. They assumed that products are interchangeable and the objective is to minimize the makespan. They present a mathematical model and propose two metaheuristics to solve the problem. Yazdani et al. (2015) study a similar problem to the one studied by Joo and Kim (2013). They propose a MIP formulation and present a metaheuristic.

Boysen (2010) studies the scheduling problem in the food industry where storage is not allowed. Also, he assume that products are not interchangeable and all trucks are available at time zero. Also, he neglect the transportation time of commodities at the terminal. He consider a discrete time window where each time slot equals 15 minuets and each truck is assigned to one time slot. Also, he assumes that trucks can be loaded/unloaded in one time slot. He considers three objective functions to minimize the flow time, the processing time and the tardiness. He proposes a mathematical formulation and develops an exact method based on the dynamic programming and a simulated annealing algorithm.

2.3 Multiple Docks With Internal Transshipment Time

Guo et al. (2012) study multi-objective dock to door scheduling problems. They considered two objective functions. In the first, they minimize the transportation cost and in the second, they minimize the weighted earliness and tardiness. They assumed that the truck arrival time and due dates are known, the preemption is not allowed, and products are not interchangeable. To solve the problem, they presented a genetic algorithm and local search procedures.

Van Belle et al. (2013) assume that the arrival and departure time of trucks are known and the truck change over time is fixed. Also, they assume that the transportation of products within the terminal is a function of distance and the speed is constant. Their objective is to minimize the weighted sum of the total transportation time and total tardiness. They present a MIP formulation and propose a tabu search heuristic and local search to solve the problem. A similar work is done by Miao et al. (2014). The differences are that, in the latter, the truck scheduling time is fixed, but the assignment of trucks to doors is a decision to be made. Also, unfulfilled shipments are allowed and the objective function of the latter is to minimize to the total operation cost and the total penalty cost for the unfulfilled shipments. They present a MIP formulation and to solve the problem, they present an adaptive tabu search algorithm.

Assadi and Bagheri (2016) and Wisittipanich and Hengmeechai (2017) assume that products are interchangeable. The former minimizes the total weighted earliness and tardiness. Also, they presented a MIP formulation and propose two metaheuristics algorithms to solve the problem. The latter minimizes the makespan. They present a MIP formulation and propose the particle swarm optimization algorithm to solve the problem.

Kuo (2013) assumes that the sequence at which trucks are served are predetermined, trucks are available at time zero, each product needs one unit of time to unload or load, and preemption is not allowed. Also, the transfer time between inbound and outbound doors is a function of the rectilinear distance between doors, and the speed is fixed for all products. The makespan is calculated after assigning trucks to doors and the objective is to minimize the makespan. They propose a model to calculate the makespan, variable neighborhood search algorithm, and four simulated Annealing algorithms.

Hermel et al. (2016) study a hierarchical approach for the truck to door scheduling problem. They assumed that all trucks are available at time zero, preemption is not allowed, products are not interchangeable, and resources at the terminal are limited. The objective is to minimize the makespan. They decompose the problem into four sub problems and solve them in a given sequence to get the final trucks assignment and schedule at doors. The four problems include clustering of trucks, assigning trucks to doors, allocating resources and scheduling trucks.

Shakeri et al. (2012) assume that trucks are loaded with standard sized pallets, and the position or the order in which pallets are unloaded or loaded are known in advance. Unloading or loading each pallet needs one unit of time and one pallet is moved at a time by a forklift. They assume that resources, such as manpower and forklifts at the cross-dock terminal, are limited. They present a MIP formulation, where the objective to minimize the makespan. They propose two phases heuristic to solve the problem. The first phase, sequences or orders the trucks and the second, assigns the trucks to the doors. Shakeri et al. (2016) propose a new solution method for the same problem. They combined the incremental neighborhood evaluation with two metaheuristics: tabu search and variable neighborhood search. Ye et al. (2018) study a similar problem, but they considered an exclusive mode terminal. They present a MIP formulation and solve it with a particle swarm optimization algorithm. They compare the performance of the particle swarm optimization and Genetic algorithms. They conclude that the particle swarm optimization algorithm provides a better solution in less time.

Chapter 3

Problem Definition

In this chapter, we formally define the TDSP. Then, we propose two MIP formulations.

3.1 Problem Description

Let M and N denote the set of inbound trucks and the set of outbound trucks. Let I denotes the set of inbound doors, and J denotes the set of outbound doors. Let K denotes the set of commodities, where each commodity is originated from inbound truck $o(k) \in M$ and has outbound truck $d(k) \in N$ as its destination. For each $k \in K$, We denote w_k as the amount of commodity to be transshipped from inbound truck $o(k)$ to outbound truck $d(k)$. For each pair of inbound door $i \in I$ and outbound door $j \in J$, let d_{ij} denote the distance or travel time required to transfer a unit of commodity from inbound door i to outbound door j. Let T denotes the set of time slots representing the planning horizon. The unloading and loading process time for each inbound and outbound truck is denoted by p_m and p_n. The time needed to transfer commodity k from inbound door i to outbound door j is $w_k d_{ij}$.

We assume that all trucks are available at time zero, ready for loading and unloading and there is no due date. Also, we assume that the parameters w_k, d_{ij}, p_m and p_n are constant and known in advance. We assume the preemption is not allowed and once loading or unloading of a truck is started, it is necessary to keep the truck at the door till the end of loading or unloading time. For the door environment, we consider an exclusive mode, which means a door is predetermined to be either an inbound door or an outbound door. Also, products are assumed to be non-interchangeable. In other words, each commodity is coming from a specific inbound truck is assigned to a specific outbound truck. Once all commodities from an inbound truck are unloaded, they are sorted, consolidated, and transferred to their respective outbound doors at the terminal. We assume doors are identical, which means that assigning a truck to any door results in the same process time. For an outbound truck, all commodities associated with it must be ready and available at the door before the truck is assigned to that door. We discretize the planning horizon and a truck can only be assigned at the beginning of each $t \in T$. Each door can process one truck at a time, which means that at each door at each time slot one truck could be served at most. We assume that doors are uncapacitated, in terms of the number of trucks they can serve. Also, we assume the time horizon is enough to process all the trucks.

The *Truck to Door Scheduling Problem* (TDSP) aims to assign trucks to doors and to determine the schedule of trucks at each door. Each inbound/outbound truck must be assigned to an inbound/outbound door. Also, at each door, trucks are scheduled such that there is no conflict in the schedule. The objective functions is to minimize the makespan. That is, the objective is to minimize the departure time of the last outbound truck leaving the cross-dock.

3.2 Mathematical Programming Formulation

In this part, we introduce two mathematical programming formulations for the TDSP. To formulate the TDSP problem, we use three sets of binary decision variables. The first set is to assign and schedule the inbound trucks at the inbound doors. For each inbound truck $m \in M$, for each inbound door $i \in I$, and for each time slot $t \in T$ we define:

$$x_{mit} = \begin{cases} 1 & \text{if inbound truck } m \text{ is assigned to inbound door } i \text{ at time } t. \\ 0 & \text{otherwise.} \end{cases}$$

For each outbound truck n, for each outbound door j, and for each time slot t we define:

$$y_{njt} = \begin{cases} 1 & \text{if outbound truck } n \text{ is assigned to outbound door } j \text{ at time } t. \\ 0 & \text{otherwise.} \end{cases}$$

The third set variables are commodities path selection variables. For each commodity, these variables link the origin and the destination of the commodity to the doors they are assigned to. For each commodity k, inbound door i and outbound door j we define:

$$z_{kij} = \begin{cases} 1 & \text{if commodity } k \text{ transits via inbound door } i \text{ and outbound door } j. \\ 0 & \text{otherwise.} \end{cases}$$

$$
\begin{aligned}
\text{minimize}\quad & C_{max} \\
\text{subject to:}\quad & \sum_{j\in J}\sum_{t\in T}(t+p_n)y_{njt} \le C_{max} && \forall n \in N && (1)\\
& \sum_{i\in I}\sum_{j\in J} z_{kij} = 1 && \forall k \in K && (2)\\
& \sum_{i\in I}\sum_{t\in T} x_{mit} = 1 && \forall m \in M && (3)\\
& \sum_{j\in J}\sum_{t\in T} y_{njt} = 1 && \forall n \in N && (4)\\
& \sum_{j\in J} z_{kij} = \sum_{t\in T} x_{o(k)it} && \forall k \in K, i \in I && (5)\\
& \sum_{i\in I} z_{kij} = \sum_{t\in T} y_{d(k)jt} && \forall k \in K, j \in J && (6)\\
& \sum_{m\in M}\sum_{a\in[max\{0,t-p_m+1\},t]} x_{mia} \le 1 && \forall t \in T, i \in I && (7)\\
& \sum_{n\in N}\sum_{a\in[max\{0,t-p_n+1\},t]} y_{nja} \le 1 && \forall t \in T, j \in J && (8)\\
& \sum_{j\in J}\sum_{t\in T} t y_{d(k)jt} - \sum_{i\in I}\sum_{t\in T} t x_{o(k)it} \ge \\
& \qquad \sum_{i\in I}\sum_{j\in J}(p_m + w_k d_{ij}) z_{kij} && \forall k \in K && (9)\\
& x_{mit} \in \{0,1\} && \forall m \in M, i \in I, t \in T && (10)\\
& y_{njt} \in \{0,1\} && \forall n \in N, j \in J, t \in T && (11)\\
& z_{kit} \ge 0 && \forall k \in k, i \in I, j \in J && (12)
\end{aligned}
$$

The objective function seeks to minimize the makespan. Constraints (1) computes the makespan value by finding the last scheduled truck. It ensures that the makespan should be grater or equal to the departure time of each outbound truck. Constraints (2) ensure that each commodity is transmitted via a single pair of inbound & outbound doors. Constraints

(3) ensures that each inbound truck is being assigned to a single inbound door at a single time slot. Similarly, constraints (4) ensure that each outbound truck is being assigned to a single outbound door at a single time slot. Constraints (5) state that if the origin of commodity k is assigned to inbound door i, then commodity k must transmit via inbound door i. At the outbound side, (6) state that if the destination of commodity k is assigned to outbound door j, then commodity k must transmit via outbound door j. Constraints (7) and (8) state that each door cannot serve more than one truck at a time and there is no conflicts in the schedule at each doors. Constraints (9) ensure there is enough time to unload and transfer all arrived commodities from inbound trucks before assigning outbound trucks to doors.

Now, we propose the second formulation, that does not require the set of z_{kij} variable. The formulation eliminates the set of z_{kij} variable in the cost of adding a huge number of constraints. We see the impact of that in the computational results section, where we provide a comparison between the two formulations.

minimize C_{max}

subject to: (1), (3), (4), (7), & (8)

$$y_{d(k)jt} \leq \sum_{i \in I} \sum_{s \in S_{kijt}} x_{o(k)is} \qquad \forall k \in K, j \in J, t \in T \tag{13}$$

$$x_{mit} \in \{0, 1\} \qquad \forall m \in M, i \in I, t \in T \tag{14}$$

$$y_{njt} \in \{0, 1\} \qquad \forall n \in N, j \in J, t \in T \tag{15}$$

Where,

$$S_{kijt} := \{s \in \mathbb{Z}^+ : 0 \leq s \leq t - \lceil p_{o(k)} + w_k d_{ij} \rceil\} \tag{16}$$

Constraints (13) are equivalent to (12) from the previous formulation. This constraint states that, the destination of commodity k can be assigned to door j at time t, if and only if, its origin is scheduled at a door such that there is enough time to unload and transfer the commodity from the origin truck to the outbound loading area.

Moreover, we try to take advantage of both formulations. To solve a MIP model, one can use many commercial and open source solvers. Often, the Branch & Bound enumeration tree is the base for these solvers. Indeed, the quality of the lower bound obtained at each node in the B&B enumeration tree directly impacts the performance of theses solvers. In a simple word, a better the LP bound results in a better performance for the solver in term of computational time. We improve lower bound results from the LP relaxation of F(1) by adding constraints (13) from F(2) to F(1). We add these constraints as valid inequalities. However, adding these constraints to the formulation results in adding a massive number of constraints. Also, not all of these constraints are necessary to obtain the optimal solution. To solve the new formulation and handle the constraints efficiently, we develop a Branch and Cut based algorithm. In this algorithm, we add a subset of constraints (13) at some nodes in the enumeration tree.

At the root node, we solve the LP relaxation of formulation F(1). After obtaining a solution, we evaluate the solution in constraint (13), which may result in the set of violated constraints. For each commodity, the algorithm adds the most violated constraint. We find the most violated constraints by inspection. Moreover, the algorithm generates the cuts at the root node and every node at every third level in the branch & bound tree. To add the constraint, we used user Callback function in CPLEX, and we add them as purge cuts. At each iteration, the algorithm adds a subset of violated cuts. At the root node, we add at most sets of 40 cuts, and if there is no improvement, we stop after adding 20 sets of cuts. At any node other than the root node at most two sets of constraints are added. The reason to limit the number of cuts to be added is to avoid increasing the size of the problem which results in more computational time at each node.

Chapter 4

Solution Algorithms for the TDSP

In this section, we present several methods to solve the TDSP. We start by presenting a set of dispatching rules we use to develop various constructive heuristics. We then present a local search algorithm that can be used to improve the initial solution obtained with this constructive heuristics. We also combine the dispatching rules, constructive algorithms, and the local search algorithm to develop two metaheuristics: an iterated local search ILS and a greedy randomized adaptive search procedure GRASP. Finally, we present the results of computational experiments and perform a set of sensitivity analysis. In the computational experiments section, we compare the results with CPLEX using the two formulations, presented in Chapter 3, as well as the results from the proposed solution methods.

4.1 Heuristics Algorithms

In this section, we present solution algorithms that provide feasible solutions in a short amount of time. We adopt the use of dispatching rules which is a conventional technique for solving scheduling problems. Dispatching rules are used in developing heuristic algorithms and usually, they are the first step performed in the heuristics. Indeed, optimality of solutions generated from the heuristics are not guaranteed, but good feasible solutions are obtained in a short amount of time for large size instances. We start by presenting a set of dispatching rules. We also present a composite dispatching rule that combines several dispatching rules. Once dispatching rules are defined, we use them to implement two constructive algorithms. We then define some neighborhoods to be used in variable neighbourhood decent VND methods.

4.1.1 Dispatching Rule

In the context of machine scheduling problems, dispatching rules are frequently used to prioritize and order a set of jobs that needed to be scheduled on one or machines. These rules are based on the attribute of the jobs and machines such as processing time, release date, due date, etc. Dispatching rules are well-known in the industry and have been extensively studied in the literature. Panwalkar and Iskander (1977) establishes a summary of more than a 100 such rules. Some of the dispatching rules result in an optimal solution for specific variant of machine scheduling problems. Composite dispatching rules combine more than one dispatching rule for prioritizing jobs. Dispatching rules are commonly used to design heuristic algorithms.

In this section, we present six dispatching rules to prioritize and order trucks. These dispatching rules rank trucks using one of their attributes such as processing time, amount of commodity, etc. Then, the dispatching rule sorts trucks in nonincreasing or nondecreasing order. We finally present a composite dispatching rule which combines diffident dispatching rules. Using three attributes of trucks, we define six dispatching rules. The first two are the classical longest processing time first (LPT) and the shortest processing time first (SPT) rules. These rules order trucks based on the processing time required to load or unload them. The next two rules are the highest number of linked trucks first (HNLT) and the fewest number of linked trucks first (FNLT). A pair of inbound truck

$m \in M$ and an outbound truck $n \in N$ are linked if there is a commodity $k \in K$ such that $o(k) = m$ and $d(k) = n$. The number of the linked trucks to a given inbound truck m is $|L_m|$ where $L_m = \{n \in N : \exists k \in K, o(k) = m, d(k) = n\}$. On the other hand, the number of the linked trucks to a given outbound truck $n \in N$ is $|L_n|$ where $L_n = \{m \in M : \exists k \in K, o(k) = m, d(k) = n\}$. In HNLT and FNLT, we order trucks based on the number of linked trucks. Finally, the last two rules are the longest total processing time (LTPT) and the shortest total processing time (STPT). The total processing time is $TP_m = \sum_{n \in \bar{L}_m} p_n$ for an inbound truck m and $TP_n = \sum_{m \in \bar{L}_n} p_m$ for an outbound truck n. In other words, the total processing time of an inbound truck is the summation of the process time of the linked outbound trucks. In these two rules, we order trucks based on the total processing time.

We can classify these six dispatching rules based on the way trucks are ordered (i.e nonincreasing, nondecreasing). LPT, HNLT, and LTPT orders trucks in nonincreasing order according to their attributes values. On the other hand, SPT, FNLT, and STPT orders trucks in nondecreasing order according to their attributes values. In the next section, we employ these dispatching rules to design deterministic constructive algorithms. The results of Section 4.3, show that dispatching rules that orders truck in nonincreasing way outperforms their opposite, in terms of quality of the obtained solution. For this reason, we present the composite dispatching rule that combines dispatching rules of type high first only.

A composite dispatching rule combines two or more dispatching rules. We now present a method for combining three dispatching rules: LPT, HNLT, and LTPT. For each truck, we assign a ranking index value by using three attributes of trucks: process time, number of linked trucks, and the total process time of linked trucks. Then, we order trucks in a nonincreasing order based on their ranking index value. So, for each inbound truck $m \in M$ and outbound truck $n \in N$, let RI_m^{in} and RI_n^{out} denote the ranking index value. Let

A_m and A_n denote the set of attributes of inbound truck m and outbound truck n where, $A_m = \{p_m, |L_m|, TP_m\}$ and $A_n = \{p_n, |L_n|, TP_n\}$. $a_m(i)$ and $a_n(i)$ denote the value of attribute i for inbound truck m and outbound truck n. Also, $a^{in}_{max}(i)$ and $a^{out}_{max}(i)$ are the maximum values of attribute i for all trucks at each side. Finally, K_i is a controlling parameter for attribute i. The value of RI^{in}_m and RI^{out}_n for inbound and outbound truck is defined as follows:

$$RI^{in}_m = \prod_{i \in A_m} \exp\left(\frac{a_m(i) - a^{in}_{max}(i)}{K_i a_m(i)}\right) \tag{17}$$

$$RI^{out}_n = \prod_{i \in A_n} \exp\left(\frac{a_n(i) - a^{out}_{max}(i)}{K_i a_n(i)}\right) \tag{18}$$

There are three beneficial aspects of these functions. First, they allow us to integrate the three dispatching rules. The second is that these functions allow us to control the weight of each attribute to the ranking index value by using the controlling parameter K_i. To make attribute i less effective we increase the value of K_i and vice versa. The last reason is that the range of the function is from zero to one, which simplifies the analysis. These dispatching rules are used as building blocks for developing the heuristics and metaheuristics algorithms presented in the coming sections.

4.1.2 Constructive Phase

Let S_{in} and S_{out} denote sets of partial schedules for inbound and outbound trucks. Also, let $(m, i, t, r) \in S_{in}$ denote an element of partial feasible solution S_{in}. In particular, the tuple (m, i, t, r) denotes that inbound truck m is scheduled at inbound door i at time slot t in position r. Similarly, $(n, j, t, r) \in S_{in}$ denotes that outbound truck n is scheduled at outbound door j at time slot t in position r. Let $C^{out}_i(S_{in})$ and $C^{out}_j(S_{out})$ denote the makespan of inbound door i and outbound door j for a given S_{in} and S_{out} respectively. Let $R_i^{in}(S_{in})$ and $R_j^{out}(S_{out})$ denote the number of trucks assigned to inbound door i and outbound door j. As mentioned, the first step of the algorithm is to order trucks according to one of dispatching rules presented in Section 4.1.1. The dispatching rules transform M, N and L_n, for each $n \in N$, into ordered sets $\bar{M}$, $\bar{N}$ and $\bar{L_n}$. Let m_r denotes the truck ordered at the r^{th} position in $\bar{M}$ and $\bar{L_n}$. Similarly, let n_r denotes the truck ordered at the r^{th} position in $\bar{N}$. For each inbound door i, let $\bar{d_i} = \frac{\sum_{j \in J} d_{i.j}}{|J|}$ denotes the average distance from inbound door i to outbound door. Let $st_m^{in}(S_{in})$ and $st_n^{out}(S_{out})$ be the starting time of inbound truck m and outbound truck n given by the partial solution. Let $f(S_{in}, S_{out})$ denotes the makespan of schedules S_{in} and S_{out}.

We now present two constructive algorithms to obtain an initial feasible solution. We refer to them as the sequential schedule constructive (SSC) heuristic and integrated schedule constructive (ISC) heuristic. In the former one, the algorithm independently schedules first inbound trucks and then schedules outbound trucks whereas, in the latter, the algorithm schedules inbound tucks and outbound trucks simultaneously at each iteration. Both algorithms start by ordering trucks according to one of the dispatching rules presented in Section 4.1.1. These dispatching rules provide an initial ordering of the set of trucks and schedule trucks to doors based on such ordering. Both algorithms start with an empty solution and at each iteration, they schedule one truck to build up a feasible solution.

The pseudo code for SSC algorithm is depicted in Algorithm 1. The algorithm starts by ordering trucks according to a given dispatching rule. It then schedules inbound trucks at inbound doors. The algorithm iterates over positions of the set $\bar{M}$. At iteration r, the algorithm finds an inbound door that results in the minimum estimated makespan for inbound truck m_r. To do that, the algorithm iterates over inbound doors $i \in I$ and computes the estimated makespan, that is $C_{i^*}^{in}(S_{in}) + \bar{d_{i^*}}(\sum_{n \in N} w_{m_r,n})$. Then, it schedules inbound truck m_r at the next available position at the door with the minimum estimated makespan. Next, the algorithm updates the inbound schedule. The algorithm continues until it schedules all inbound trucks.

Once all inbound truck have been schedule, the algorithm starts to schedule outbound trucks at outbound doors. The algorithm iterates over the positions of the order set $\bar{N}$. At each iteration, for outbound truck n_r, it finds an outbound door that results in the earliest starting time. For that, the algorithm iterates over $j \in J$ and obtains the earliest starting time $EST^{out}_{n_r,j}(S_{in}, S_{out})$ for outbound truck n_r at outbound door j. The earliest starting time of a truck at a door is the maximum of the doors makespan or the time when all commodities associated with such outbound truck are available for loading. That is, $EST^{out}_{n_r,j}(S_{in}, S_{out}) = \min_{t \in R^+}\{\, t | t \geq C^{out}_j(S_{out}),\, t \geq st^{in}_m(S_{in}) + p_{m_r} + d_{i^*j} w_{m_r,n} \forall m \in L_n\}$. Then, the algorithm selects outbound door j^* with minimum the $EST(n_r, j)$ and adds the new element $\big(n_r, j^*, EST^{out}_{n_r,j^*}(S_{in}, S_{out}), R^{out}_{j^*}(S_{out})\big)$ to S_{out}. Once all trucks have been scheduled, the algorithm returns S_{in} and S_{out}, which correspond to a feasible solution.

Algorithm 1 sequential constructive algorithm SSC (DR)

```
Inputs $M$, $N$, $p$, $d$, $w$, $L_n$, DR.
set $\bar{M}$ according to dispatching order DR
set $\bar{N}$ according to dispatching order DR
for $r = 1; r \leq |M|; r++$ do
    $i^* \leftarrow \arg\min_{i \in I}\{\, C^{in}_i(S_{in}) + \bar{d}_i * \sum_{n \in L_{m_r}} w_{m_r,n}\}$
    $S_{in} \leftarrow S_{in} \bigcup \big(m_r, i^*, C^{in}_{i^*}(S_{in}), R(i^*)\big)$
end for
for $r = 1; r \leq |N|; r++$ do
    $j^* \leftarrow \arg\min_{j \in J}\{\, EST^{out}_{n_r,j}(S_{in}, S_{out})\}$
    $S_{out} \leftarrow S_{out} \bigcup \big(n_r, j^*, EST^{out}_{n_r,j^*}(S_{in}, S_{out}), R^{out}_{j^*}(S_{out})\big)$
end for
Return $S_{in}, S_{out}$;
```

ISC algorithm schedules inbound and outbound trucks simultaneously. Let S^{temp}_{in} and S^{temp}_{out} denote temporary copies of partial solutions S_{in} and S_{out}. As illustrated in Algorithm 2, it starts by ordering the set of outbound trucks N and sets of linked trucks L_n according to a given dispatching rule. Then, the algorithm iterates over positions of $\bar{N}$. At iteration r_1, for each outbound $j \in J$, we obtain the earliest starting time $EST^{out}_{n_r,j}(S^{temp}_{in}, S_{out})$. For each outbound truck n_{r_1} and outbound door j, the algorithm creates a temporary copy of the partial inbound solution. Then, the algorithm schedules inbound trucks $m \in L_n$ by iterating over positions of the ordered set $\bar{L_n}$.

Algorithm 2 integrated schedule constructive ISC algorithm

```
Inputs $M$, $N$, $p$, $d$, $w$, $L_n$, Dispatching Rule.
$S_{in} \leftarrow \emptyset, S_{out} \leftarrow \emptyset$
set $\bar{N}$ according to dispatching order DR
For each $n \in N$, set $\bar{L_n}$ according to dispatching order DR
for $r_1 = 1; r_1 \leq |N|; r_1 + +$ do
    for $j \in J$ do
        $S_{in}^{temp} \leftarrow S_{in}$
        for $r_2 = 1; r_2 \leq |L_{n_{r_1}}|; r_2 + +$ do
            if $(status(S_{in}, m_{r_2}) = 0)$ then
                Select inbound door $i^*$ according to rule (a) and (b);
                $S_{in}^{temp} \leftarrow S_{in}^{temp} \bigcup \big\{ \big(m_r, i^*, C_{i^*}^{in}(S_{in}^{temp}), R_{i^*}^{in}(S_{in}^{temp})\big) \big\}$
            end if
        end for
        $t_j \leftarrow EST_{n_r,j}^{out}(S_{in}^{temp}, S_{out})$
    end for
    $j^* \leftarrow \underset{j \in J}{\text{argmin}}\{t_j\}$
    for $r_2 = 1; r_2 \leq |L_{n_{r_1}}|; r_2 + +$ do
        if $(status(S_{in}, m_{r_2}) = 0)$ then
            Select inbound door $i^*$ according to rule (a) and (b);
            $S_{in} \leftarrow S_{in} \bigcup \big\{ \big(m_r, i^*, C_{i^*}^{in}(S_{in}), R_{i^*}^{in}(S_{in})\big) \big\}$
        end if
    end for
    $S_{out} \leftarrow S_{out} \bigcup \big\{ \big(n, j, \text{EST}_{n_r,j^*}^{out}(S_{in}, S_{out}), R_j^{out}(S_{out})\big) \big\}$
end for
Return $S_{in}, S_{out}$;
```

Let $status(S, m_{r_2}) = 0$ if inbound truck m_{r_2} is not scheduled yet. At iteration r_2, if inbound truck m_{r_2} is not scheduled yet in S_{in}, the algorithm schedules the inbound truck at an inbound door i^* and updates the temporary solution S_{in}^{temp}. We consider the following two rules for selecting i^*:

(a) $i^* \leftarrow \arg\min_{i \in I}\{ d_{ij} | C_i^{in}(S_{in}^{temp}) + p_{m_{r_2}} + d_{ij} w_{m_{r_2},n_{r_1}} \leq C_j^{out}(S_{out}^{temp})\}$

(b) $i^* \leftarrow \arg\min_{i \in I}\{ C_i^{in}(S_{in}^{temp}) + p_{m_r} + d_{ij} w_{m_r,n}\}$

We apply rule (a) if there exists an inbound door which results in commodities waiting for outbound truck n_{r_1}. That is, commodities from m_{r_2} arrive at outbound door j before outbound door j becomes available. In the (a), we select the nearest inbound door satisfying the condition $C_i^{in}(S_{in}^{temp}) + p_{m_{r_2}} + d_{ij}w_{m_{r_2},n_{r_1}} \leq C_j^{out}(S_{out}^{temp})$. In the other hand, if there is no such an inbound door, we apply rule (b) and schedule the inbound truck at the inbound door with minimum $C_i^{in}(S_{in}^{temp}) + p_{m_r} + d_{ij}w_{m_r,n}$. Once all trucks in $L_{n_{r_1}}$ have been scheduled, the algorithm obtains the earliest starting time $EST_{n_r,j}^{out}(S_{in}^{temp}, S_{out})$ and stores its value in t_j.

For inbound truck n_{r_1}, once we obtain t_j for all outbound doors, the algorithm selects outbound door j^* corresponding to the earliest starting time. Then, for all inbound trucks in L_{n_1}, the algorithm schedules inbound truck by selecting inbound door i^* according to the two rules mentioned above. After that, the algorithm obtains the earliest starting time and updates the partial solutions. The algorithm terminate once all trucks have been scheduled.

4.1.3 Local Search

Local search heuristics improve feasible solutions by exploring its predefined neighborhood. In this section, we start by presenting six neighbourhoods: three are based on a swap procedure, and three are based on a shift procedure. We generate the swap neighborhood by selecting one or more pairs of trucks and then interchange their doors and positions at the door. We generate inbound swap, outbound swap, and double swap neighborhoods. The second class of the neighborhood is the shift neighborhood. Here, we generate the inbound shift, outbound shift, and double shift neighborhoods. Since these movements effect the starting time of trucks, we present a procedure to reset the starting time of each truck. Furthermore, we present several restricted versions of these neighborhoods. The motivation behind reducing the size of neighborhoods is mainly to reduce the computational time. Finally, we present various variable neighborhood descent VND methods.

The inbound swap neighborhood $N^M_{swap}(S)$ is defined by selecting a pair of inbound trucks $m_1, m_2 \in M$ and then interchanging doors and positions at which they are scheduled. For instance, consider (m_1, i_1, r_1) and (m_2, i_2, r_2) which state that, the inbound truck m_1 is at the r_1^{th} position at door i_1 and the inbound truck m_2 is at the r_2^{th} position at inbound door i_2. After performing a swap procedure between m_1 and m_2 the result would be (m_1, i_2, r_2) and (m_2, i_1, r_1). Note that we can perform the swap procedure in $|M| \times (|M| - 1)$ different ways but there are $\frac{|M|\times(|M|-1)}{2}$ unique solutions in $N^M_{swap}(S)$. On the other hand, the outbound swap neighborhood $N^N_{swap}(S)$ is similar to the inbound swap neighborhood except that it is performed on the outbound trucks. The double swap neighborhood $N^{MN}_{swap}(S)$ explores all solutions generated by selecting a pairs of inbound trucks and a pair of outbound trucks and then performing a swapping procedure at both sides simultaneously. Note that the double swap procedure generates $\left(\frac{|M|(|M|-1))}{2}\right) \times \left(\frac{|N|(|N|-1))}{2}\right)$ unique solutions in neighborhood $N^N_{swap}(S)$.

The inbound shift neighborhood $N^M_{shift}(S)$ is defined by selecting one inbound truck at a time and then perform a shift movement. The procedure selects an inbound truck m and then changes its schedule from inbound door i at position r to inbound door i' at position r'. The procedure could shift the position at the same door, or it could schedule it at another inbound door. The outbound shift neighborhood $N^N_{shift}(S)$ is similar to the inbound shift neighborhood except that it is performed on the outbound trucks. The double shift neighborhood $N^{MN}_{shift}(S)$ is a combination of both inbound shifting and outbound shifting movements. The double shift neighborhood explores all solutions generated by selecting a pair of inbound trucks and outbound trucks and then performing shift movements at both sides. The double shift movement explores up to $|M^2| \times |N^2|$ solutions.

The swap and shift movements transform a solution S into a new solution S' and they at least change the schedule of one truck. Which in turn affects the starting time of other trucks. To reset the starting time of trucks, we present *reset starting time(S)* subroutine illustrated in Algorithm 3. This subroutine scans and resets the starting time for all the trucks again. Let $m_{i,r}$ and $n_{j,r}$ denote the r^{th} inbound truck and outbound truck at inbound door i and outbound door j under the schedule. The procedure starts with resetting the time index for the inbound trucks and then it resets the time index of the outbound trucks.

As illustrated in Algorithm 3, the subroutine iterates over each inbound door $i \in I$. At each inbound door i, the subroutines rests the starting time of inbound truck according to the order defined by the schedule. After that, the subroutine rests the starting time of the outbound trucks by iterating over each outbound truck and each truck scheduled at that door. At each iteration the subroutine updates the starting time of each outbound truck and the makespan of each door. We note that the complexity of resetting and scanning in the worst case is $O(|N| + |N| \times |M|)$. So, the worst case running time required to explore all solutions in inbound shift neighborhood $(N^M_{shift}(S))$ is $O((|N| + |N| \times |M|)|M|^2)$. Therefore, exploring a subset of the neighborhood could be better in terms of computational time.

Algorithm 3 reset starting time(S)

```
for $(i = 1; i \leq |I|; i++)$ do
    $C(i) = 0;$
    for $(r = 1; r \leq R(i); r++)$ do
        $st(m_{i,r}) = C_{in}(i);$
        $C(i) = st(m_{i,r}) + p_{m_{i,r}};$
    end for
end for
for $(j = 0; j < |J|; j++)$ do
    $C(j) = 0;$ EST = 0;
    for $(r = 0; r < R(j); r++)$ do
        for $(m \in L_{n_{j,r}})$ do
            EST = max$\{EST, C(j), st(m) + p_m + w_{m,n_{j,r}} * d_{dr(m),j}\};$
        end for
        $st(n_{j,r}) = max :$
        $C(j) = st(n_{j,r}) + p_n;$
    end for
end for
Return S;
```

We now present two strategies to reduce the size of a neighborhood and then we use them to define seven restricted versions of the neighborhoods presented before. Recall that, the neighbourhood of a solution is explored by executing a swap, shift, double swap or double shift movements. The first strategy is to focus only on the movements such that the distance between the original door and the door at which the truck is swapped to (or shift) to is less than some threshold value. Using this method we present a restricted version of $N^{MN}_{swap}(S)$ and $N^{MN}_{shift}(S)$. The second method is to define a set of trucks and then only evaluate the neighbors associated with such subset. Also, we combine these two strategies to develop restricted versions of $N^{M}_{swap}(S)$, $N^{N}_{swap}(S)$, $N^{M}_{shift}(S)$, $N^{N}_{shift}(S)$, and $N^{MN}_{swap}(S)$.

We now present restricted versions of the two neighbourhoods $N^{MN}_{swap}(S)$ and $N^{MN}_{shift}(S)$ denoted as $N^{MN}_{swap}(S,\Delta)$ and $N^{MN}_{shift}(S,\Delta)$, which are restricted by using the first strategy. These restricted neighbourhoods define solutions that we can obtain by performing a shift or a swap procedure at both sides, only if the distance between the original and new door is less than Δ. Let $dr(m)$ denote the inbound door where inbound truck m is scheduled and $dr(n)$ denote outbound door for outbound truck n as well. Let $D(i,i')$ and Let $D(j,j')$ denotes the distance between two inbound doors $i,i' \in I$ and the distance between two outbound doors $j,j' \in J$. We define a controlling parameter Δ to control size of a restricted neighborhood. $N^{MN}_{swap}(S,\Delta)$ defined by selecting any pair of inbound truck $m_1,m_2 \in M$ and outbound truck $n_1,n_2 \in N$. If $D(dr(m_1),dr(m_2))$ and $D(dr(n_1),dr(n_2))$ is less than Δ, then we swap m_1 with m_2 and n_1 with n_2. Otherwise, we do not preform the swap movement. On the other hand, $N^{MN}_{shift}(S,\Delta)$ is defined by selecting any pair of inbound truck $m \in M$ and outbound truck $n \in N$. Then, we shift inbound truck m from inbound door i to inbound door i' and outbound truck n from outbound truck j to outbound door j', only if $D(i,i')$ and $D(j,j')$ is less than Δ.

We now present the restricted versions of $N^{M}_{swap}(S)$, $N^{N}_{swap}(S)$, $N^{M}_{shift}(S)$, $N^{N}_{shift}(S)$, and $N^{MN}_{swap}(S)$. For that, we combine both strategies to develop restricted neighbourhoods.

Let $N^M_{swap}(S, \Delta, B)$ and $N^N_{swap}(S, \Delta, B)$ denote restricted version of $N^M_{swap}(S)$ and $N^N_{swap}(S)$, respectively. The restricted inbound swap neighborhood $N^N_{swap}(S, \Delta, B)$ is defined by selecting a pair of inbound truck $m_1, m_2 \in M$. Then, we swap m_1 with m_2, only if $D(dr(m_1), dr(m_2))$ is less than Δ and either $m_1 \in B$ or $m_2 \in B$. The restricted outbound swap neighborhood $N^N_{shift}(S, \Delta, B)$ is similar to the restricted inbound swap neighborhood except that it is performed on the outbound trucks. Moreover, let $N^M_{shift}(S, \Delta, B)$ and $N^N_{shift}(S, \Delta, B)$ denote the restricted version of $N^M_{shift}(S)$, and $N^N_{shift}(S)$, respectively. The restricted inbound shift neighbourhood $N^M_{shift}(S, \Delta, B)$ selects an inbound truck m and then shift its schedule from inbound door i at position r to inbound door i', only if $D(i, i')$ is less than Δ and $m \in B$. The restricted outbound shift neighborhood $N^N_{shift}(S, \Delta, B)$ is similar to the restricted inbound shift neighborhood except that it is performed on the outbound trucks. The restricted double swap neighbourhood $N^{MN}_{swap}(S, \Delta, B_{in}, B_{out})$ is defined by selecting a pair of inbound trucks $m_1, m_2 \in M$ and outbound trucks $n_1, n_2 \in N$. If $D(dr(m_1), dr(m_2))$ and $D(dr(n_1), dr(n_2))$ are less than Δ, either $m_1 \in B_{in}$ or $m_2 \in B_{in}$, and either $n_1 \in B_{in}$ or $n_2 \in B_{out}$, then we swap m_1 with m_2 and n_1 with n_2.

Local search is an iterative procedure that explores the neighbourhood of a solution S to search for another solution S' with better objective function value. The procedure terminates when it can not find a better solution. In this case, the solution is local optimal. We now present various local search procedures based on the variable neighborhood descent (VND) method for the TDSP. The VND is proposed by Hansen and Mladenović (2001). This method systematically explores m sets of neighbourhoods $N_1, N_2, ..., N_m$. VND explores solutions in N_1 until it finds a local optimal solution. After that, the algorithm search solutions in $N_2, ..., N_m$ in sequence. Once VND finds an improved solution it starts from N_1 again. It keeps doing that until it cannot find an improved solution. We present five versions of VND, each of which with different sets of neighbourhoods. We refer to them as VND$_1$, ..., VND$_5$. After performing preliminary computational experiments, we define each VND procedure as follows:

- VND$_1$: $N^M_{swap}(S, 6, L_n)$, $N^N_{swap}(S, L_n)$. $N^M_{shift}(S, 6, L_n)$,$N^N_{shift}(S, L_n)$, $N^{MN}_{swap}(S, 5, L_n, \{n\})$.
- VND$_2$: $N^M_{swap}(S)$, $N^M_{shift}(S)$, $N^N_{swap}(S)$, $N^N_{shift}(S)$, $N^{MN}_{swap}(S, 5)$.
- VND$_3$: $N^M_{swap}(S)$, $N^N_{swap}(S)$, $N^M_{shift}(S)$, $N^N_{shift}(S)$, $N^{MN}_{swap}(S, 5)$, $N^{MN}_{shift}(S, 0)$.
- VND$_4$: $N^M_{swap}(S)$, $N^M_{shift}(S)$, $N^N_{swap}(S)$, $N^N_{shift}(S)$.
- VND$_5$: $N^M_{swap}(S)$, $N^M_{shift}(S)$, $N^N_{swap}(S)$, $N^N_{shift}(S)$, $N^{MN}_{swap}(S, 2)$.

4.1.4 Combining Constructive and Local Search Heuristics

Now we combine constructive algorithms and VND methods to develop two heuristics algorithms. The first heuristic combines SSC, VND_1, VND_2, and VND_3. Whereas, the second combines ISC, VND_1, VND_2, and VND_3. We refer to them as the sequential schedule constructive local search (SSCLS) heuristic and integrated schedule constructive local search (ISCLS) heuristic. These two heuristics construct a feasible solution and explore the neighbourhoods to obtain a local optimal solution.

The SSELS algorithm is illustrated in Algorithm 4. This algorithm is similar to the SSC, but in SSELS we embed three local search procedures at different stages. SSELS algorithm orders trucks according to a dispatching rule and then it schedules inbound trucks. After that, for outbound truck n_r, we obtain C_j for all outbound doors. C_j stores the makespan results from scheduling outbound truck n_r at the next available position at outbound door j and then preforming $VND_1(S_{in}, S_{out})$ method. Then, we select outbound door j^* which correspond to the door with the minimum C_j and schedule the outbound truck at that door. After that, we preform $VND_2(S_{in}, S_{out})$ method to improve the partial schedule. Once all trucks have been scheduled, we perform $VND_2(S_{in}, S_{out})$ method to obtain a local optimal solution.

Algorithm 4 sequential schedule constructive algorithm local search SSCLS (DR)

```
**Inputs** $M$, $N$, $p$, $d$, $w$, $L_n$, DR.
set $\bar{M}$ according to dispatching order DR
set $\bar{N}$ according to dispatching order DR
**for** $r = 1; r \leq |M|; r++$ **do**
    $i^* \leftarrow \arg\min_{i \in I}\{\, C_i^{in}(S_{in}) + \bar{d}_i * \sum_{n \in L_{m_r}} w_{m_r,n}\}$
    $S_{in} \leftarrow S_{in} \bigcup \big(m_r, i^*, C_{i^*}^{in}(S_{in}), R(i^*)\big)$
**end for**
**for** $r = 1; r \leq |N|; r++$ **do**
    **for** $j \in J$ **do**
        $C_j \leftarrow f\Big(VND_1\big(S_{in}, S_{out} \bigcup \big(n_r, j, EST_{n_r,j}^{out}(S_{in}, S_{out}), R_j^{out}(S_{out})\big)\big)\Big)$
    **end for**
    $j^* \leftarrow \arg\min_{j \in J}\{C_j\}$
    $S_{in}, S_{out} \leftarrow VND_2\big(S_{in}, S_{out} \bigcup \big(n_r, j^*, EST_{n_r,j^*}^{out}(S_{in}, S_{out}), R_{j^*}^{out}(S_{out})\big)\big)$
**end for**
$S_{in}, S_{out} \leftarrow VND_3\big(S_{in}, S_{out}\big)$
**Return** $S_{in}, S_{out}$;
```

The ISCLS algorithm is illustrated in Algorithm 5. This algorithm is similar to ICS heuristic, but in ISCLS we embed various VND methods at different stages of the algorithm. It starts by ordering trucks according to a given dispatching rule. Then, the algorithm iterates over outbound truck r_1 and outbound door $j \in J$. Also, the algorithm obtains C_j which stores the makespan value of a partial solution resulting from scheduling outbound truck n_{r_1}, inbound trucks linked to n_{r_1}, and performing VND_1. Once the algorithm obtains C_j for all outbound doors, it schedules all inbound trucks that are in set $L_{n_{r_1}}$ not scheduled yet to outbound door i^* respecting role (a) and (b) presented in Section 4.1.2. Also, it schedules outbound truck n_{r_1} at outbound door j^* that corresponds to the door with minimum C_j. Following that, the algorithm performs VND_2 to obtain a local optimal partial feasible schedule. Once all trucks have been scheduled, the algorithm calls VND_3 to obtain a local optimal solution.

Algorithm 5 integrated schedule constructive local search algorithm

```
**Inputs** $M$, $N$, $p$, $d$, $w$, $L_n$, Dispatching Rule.
$S_{in} \leftarrow \emptyset, S_{out} \leftarrow \emptyset$
set $\bar{N}$ according to dispatching order DR
For each $n \in N$, set $\bar{L_n}$ according to dispatching order DR
**for** $r_1 = 1; r_1 \leq |N|; r_1 + +$ **do**
    **for** $j \in J$ **do**
        $S_{in}^{temp} \leftarrow S_{in}$
        **for** $r_2 = 1; r_2 \leq |L_{n_{r_1}}|; r_2 + +$ **do**
            **if** ($status(S_{in}, m_{r_2}) = 0$) **then**
                Select inbound door $i^*$ according to rule (a) and (b);
                $S_{in}^{temp} \leftarrow S_{in}^{temp} \bigcup \big\{ \big(m_r, i^*, C_{i^*}^{in}(S_{in}^{temp}), R_{i^*}^{in}(S_{in}^{temp})\big) \big\}$
            **end if**
        **end for**
        $C_j \leftarrow f\Big(VND_1\big(S_{in}^{temp}, S_{out} \bigcup \big(n_r, j, EST_{n_r,j}^{out}(S_{in}, S_{out}), R_j^{out}(S_{out})\big)\big)\Big)$
    **end for**
    $j^* \leftarrow \underset{j \in J}{\text{argmin}}\{C_j\}$
    **for** $r_2 = 1; r_2 \leq |L_{n_{r_1}}|; r_2 + +$ **do**
        **if** ($status(S_{in}, m_{r_2}) = 0$) **then**
            Select inbound door $i^*$ according to rule (a) and (b);
            $S_{in} \leftarrow S_{in} \bigcup \big\{ \big(m_r, i^*, C_{i^*}^{in}(S_{in}), R_{i^*}^{in}(S_{in})\big) \big\}$
        **end if**
    **end for**
    $S_{in}, S_{out} \leftarrow VND_2\big(S_{in}, S_{out} \bigcup \big(n_r, j^*, EST_{n_r,j^*}^{out}(S_{in}, S_{out}), R_{j^*}^{out}(S_{out})\big)\big)$
**end for**
$S_{in}, S_{out} \leftarrow VND_3\big(S_{in}, S_{out}\big)$
**Return** $S_{in}, S_{out}$;
```

4.2 Metaheuristics Algorithms

In this section, we present two metaheuristics algorithms to solve the TDSP. The goal of developing metaheuristics is to obtain a high-quality solution in a reasonable amount of time. The results of Section 4.3 show that SSC heuristic outperforms ISC, in terms of quality of the obtained solution. So, we develop metaheuristics using SSC and we do not develop metaheuristics using ISC. Both metaheuristics we present here combine SSC, VND_4 and VND_5. Also, both metaheuristics starts by ordering trucks according to a dispatching rule presented before. We next introduce a greedy randomized adaptive search procedure (GRASP) and then we present the iterated local search (ILS) metaheuristic.

4.2.1 GRASP Algorithms

GRASP is a multi-start metaheuristic that is applied to solve many problems including scheduling, routing, location, assignment, etc. Festa and Resende (2009) presents a survey that covers the works done from 1989 to 2008. As illustrated in Algorithm 6, GRASP starts by ordering trucks according to a given dispatching rules and also consists of two main phases a constrictive phase and a local search phase. At iteration t, the algorithm constructs a feasible solution using a randomized constructive schedule (RCS) algorithm. Then, the algorithm improves the solution using VND_5 method. After that, the algorithm updates both the controlling parameter α and the best feasible solution s. The metaheuristic repeats this procedure until it reaches the maximum number of iterations. To control the search process, we use the controlling parameters α. Increasing α increases the randomness of solutions obtained by RCS. Here, we start by an initial α, and at each iteration, we decrease the value to $\alpha * \delta$ where $\delta \in [0, 1]$. Also, if $\alpha \leq \alpha_{min}$, we reset $\alpha = \alpha_0$.

The RCS algorithm is illustrated in Algorithm 7. As in SSC algorithm, RCS independently schedules inbound trucks and then schedules outbound trucks. However, SSC selects and schedule trucks in the order defined by the dispatching rule, whereas the RCS selects a

Algorithm 6 GRASP

```
**Inputs** $M$, $N$, $p$, $d$, $w$, $L_n$, Dispatching Rule.
set $\bar{M}$ according to dispatching order DR
set $\bar{N}$ according to dispatching order DR
**for** $t = 0; \leq$ Number Of Iteration ; $t++$ **do**
    $s_0 \leftarrow$ randomized constructive schedule algorithm $(\bar{M}, \bar{N}, I, J, D, W, Pu, Pl, \alpha)$
    $s^* \leftarrow VND_5(s_0)$
    update($\alpha$, $s$)
**end for**
**Return** $S$
```

truck randomly from a restricted candidate list RCL. Let $RI^{in}_{m_r}$ and $RI^{out}_{n_r}$ denote the ranking index of inbound and outbound trucks ordered at the r^{th} position, respectively. For $\bar{M}$ and $\bar{N}$, we assume that trucks are ordered in nonincreasing order based on the ranking index value obtained using equations (17) and (18), respectively. RI^{in}_1 and $RI^{in}_{|M|}$ correspond to the maximum and minimum ranking index values for inbound trucks. Also, RI^{out}_1 and $RI^{out}_{|M|}$ correspond to the maximum and minimum ranking index values for outbound trucks.

The algorithm starts by scheduling inbound trucks at inbound doors. It first creates V, a copy of $\bar{M}$. Then, at each iteration of the while loop, the algorithm schedules an inbound truck from V. For that, it creates a RCL which includes all inbound trucks $m \in V$, if $RI^{in}_m \leq RI^{in}_{m_1} + \alpha(RI^{in}_{m_{|V|}} - RI^{in}_{m_1})$. Then, the algorithm randomly selects an inbound truck from RCL and schedules that truck at inbound door i^*. That is the door with minimum $C^{in}_i(S_{in}) + \bar{d}_i * \sum_{n \in L_m} w_{m,n}$. After that, the algorithm removes the selected truck from V and starts over until it schedules all inbound trucks.

Once all inbound trucks have been scheduled, the RCS schedules outbound trucks. It creates V, a copy of $\bar{N}$. Then, at each iteration of the while loop, it schedules an outbound truck from V. For that, the algorithm creates a RCL and then it randomly selects an outbound truck from that list. After that, it schedules the selected truck to the next available position at the outbound door j^*. which is the door that results in the minimum earliest starting time. Then, the algorithm calls VND_4 to obtain a local minimum partial feasible schedule and then it removes the selected truck from V. The algorithm continues until it schedules all outbound trucks and finally returns a feasible schedule.

Algorithm 7 randomized constructive schedule RCS algorithm (α)

```
Inputs $\bar{M}$, $\bar{N}$, $p$, $d$, $w$, $L_n$, Dispatching Rule(DR).
$S_{in} \leftarrow \emptyset$, $V \leftarrow \bar{M}$;
while $V \neq \emptyset$ do
    RCL $\leftarrow \{m \in V | RI^{in}_{m} \leq RI^{in}_{m_1} + \alpha(RI^{in}_{m_{|V|}} - RI^{in}_{m_1}\}$;
    $m \leftarrow$ randomly select a truck from RCL
    $i^* \leftarrow \arg\min_{i \in I}\{\ C^{in}_{i}(S_{in}) + \bar{d}_i * \sum_{n \in L_m} w_{m,n}\}$
    $S_{in} \leftarrow S_{in} \bigcup \big(m, i^*, C^{in}_{i^*}(S_{in}), R(i^*)\big)$
    $V = V/\{m\}$
end while
$S_{out} \leftarrow \emptyset$, $V \leftarrow \bar{N}$
while $V \neq \emptyset$ do
    RCL $\leftarrow \{n \in V | RI^{out}_{n} \leq RI^{out}_{n_1} + \alpha(RI^{out}_{n_{|V|}} - RI^{out}_{n_1}\}$
    $n \leftarrow$ randomly select a truck from RCL
    $j^* \leftarrow \arg\min_{j \in J}\{\ EST^{out}_{n,j}(S_{in}, S_{out})\}$
    $S_{in}, S_{out} \leftarrow VND_2\big(S_{in}, S_{out} \bigcup \big(n, j^*, EST^{out}_{n,j^*}(S_{in}, S_{out}), R^{out}_{j^*}(S_{out})\big)\big)$
    $V = V/\{n\}$
end while
Return S;
```

4.2.2 Iterative Local Search Algorithm

According to Glover and Kochenberger (2006), Iterated Local Search (ILS) Algorithm maintains a chain of current solutions. To escape a local optima and to obtain a new solution, ILS applies a perturbation to the current solution and then improves the solution by exploring one or more neighborhoods. At each iteration, ILS generates a new solution, and a decision should be made even to make the new solution be the current solution or ignore it. This decision is performed through a predefined Acceptance criterion. The ILS metaheuristics is illustrated in Algorithm 8. p and α are the controlling parameters. The algorithm consists of four main subroutines: construct a feasible solution, perturbation, acceptance criterion, and update the controlling parameters. In the first, a feasible solution is constructed using SSC. Then, at each iteration, the perturbation routine takes a feasible solution S^* and partially destroys it and then the routine returns a new solution S'. The acceptance criterion takes the current solution S^* and replaces it by S', if S' meets the acceptance criterion. Finally, the update function updates the controlling parameters.

Algorithm 8 Iterated Local Search

```
**Inputs** : $p$, $\lambda$, NumberOfIteration. $\bar{M}$, and $\bar{N}$
$S_0 \leftarrow$ ConstructFeasibleSolution($\bar{M}$, $\bar{N}$);
**for** t $\leq$ Number Of Iteration **do**
    $S' \leftarrow$ Perturbation($S^*$, $p$, $\bar{M}$, $\bar{N}$);
    $S^* \leftarrow$ AcceptanceCriterion($S^*, S'^*$, $\lambda$);
    update($p$, $\lambda$)
**end for**
**Return** Best Feasible Solution
```

The perturbation subroutine is illustrated in Algorithm 9. This subroutine partly destroys the feasible schedule by removing a subset of inbound and outbound trucks from the schedule and then reschedule them. The subroutine first perturbs the inbound schedule S_{in} and then it perturbs the outbound schedule S_{out}. As illustrated in the pseudo code, the subroutine generates a random set $V \subseteq M$ where the carnality of V is less than $\lceil p \times |M| \rceil$. Then, the subroutine iterates over inbound doors and positions of doors. For inbound truck $m_{i,r}$, if $m_{i,r} \in V^{in}$, then we remove $(m_{i,r}, i, st^{in}_{m}(S_{in}), r)$ from inbound schedule S_{in} and increment the counter which counts the number of removed trucks. Otherwise, we update the position of inbound truck $m_{i,r}$. Once we have been iterated over all inbound trucks, we reschedule the removed trucks from S_{in}. For that, we iterate over the positions of $\bar{M}$. At each iteration r, if inbound truck m_r is not scheduled in S_{in}, we schedule m_r at the next available position at inbound door i^* which results in minimum estimated makespan.

Once all inbound trucks have been scheduled, we perturb the outbound schedule S_{out}. For that, we first create V^{out} a set of outbound trucks to be removed from outbound schedule S_{out} which contains all outbound trucks linked to inbound trucks in V_{in}. After that, we iterate over outbound trucks scheduled at outbound doors. For outbound truck $n_{j,r}$, if the truck is in set V^{out}, we remove $(n_{j,r}, j, st_n^{out}(S_{out}), r)$ form the outbound schedule and update the counter that corresponds to the number for removed trucks. Otherwise, we reset the position of outbound truck $n_{j,r}$ in the outbound schedule. Once all outbound trucks in V^{out} have been removed, we reschedule them. For that, we iterate over positions of $\bar{N}$. At iteration r, if outbound truck n_r is not scheduled in S_{out}, we schedule the truck at outbound door j^* which corresponds to the outbound door with the earliest starting time. We then call VND_4 method to obtain a local optimal partial schedule. Once all outbound trucks have been scheduled we call VND_5 to obtain a local minimum schedule.

The acceptance criterion procedure is presented in Algorithm 10. The controlling parameter λ controls the frequency of replacing S^* by the new solution obtained S'. Increasing λ results in fewer times S is replaced and vice versa. At each iteration, a perturbation function is applied on the solution S^* and a new solution S' is obtained. The acceptance criterion function decides whether to replace the S^* with S' for the next iterations or not. As illustrated, if $f(S') < \alpha f(S^*)$ then, S^* is replaced with S'. otherwise, the function returns S^*, the solution before applying the perturbation function.

To control the searching process, we update the controlling parameters p and λ. The first controls the strength of the perturbation function. Increasing p results in more change to the current solution. Its natural to start with small p and increase it to $p * (1 + \delta)$ where $\delta \in [0, 1]$. In the case of $p*(1+\delta) \geq p_{max}$, we set $p = p_o$. The second controlling parameter controls how often the new solution replaces the current solution. To control that, we start by an initial λ, at each iteration we decrease the value to $\lambda * \delta$ where $\delta \in [0, 1]$, and if $\lambda * \delta \leq \lambda_{min}$, we set $\lambda = \lambda_0$.

4.3 Computational Results

For the experiments, we are using Intel(R) Xeon(R) V3 with 3.10GHz and 520GB of RAM. All algorithms are coded in C and to solve the associated MIP we use the callable

Algorithm 9 Perturbation

```
Inputs: $S_{in}$, $S_{out}$, $p$, $\bar{M}$, $\bar{N}$, $L_n$, $W$, $D$, $I$, $J$.
generate random set $V^{in} \subseteq M$, such that $|V^{out}| = \lceil p \times |M| \rceil$;
for $i \in I$ do
    count = 0
    for $r = 1; r \leq R_i^{in}(S_{in}); r++$ do
        if $m_{i,r} \in V^{in}$ then
            $S_{in} \leftarrow S_{in} \backslash \big(m_{i,r}, i, st_m^{in}(S_{in}), r\big)$
            count++
        else
            $S_{in} \leftarrow S_{in} \backslash \big(m_{i,r}, i, st_m^{in}(S_{in}), r\big) \bigcup \big(m_{i,r}, i, st_m^{in}(S_{in}), r - \text{count}\big)$
        end if
    end for
end for
for $r = 1; r \leq |M|; r++$ do
    if $status(S_{in}, m_r) = 0$ then
        $i^* \leftarrow \arg\min_{i \in I}\{\ C_i^{in}(S_{in}) + \bar{d}_i * \sum_{n \in L_{m_r}} w_{m_r,n}\}$
        $S_{in} \leftarrow S_{in} \bigcup \big(m_r, i^*, C_{i^*}^{in}(S_{in}), R(i^*)\big)$
    end if
end for
generate a set $V^{out} \subseteq N$, where $n \in V^{out}$ if $\exists m \in L_n$ and $m \in V^{in}$
for $j \in J$ do
    count = 0
    for $r = 1; r \leq R_j^{out}(S_{out}); r++$ do
        if $n_{j,r} \in V^{out}$ then
            $S_{out} \leftarrow S_{out} \backslash \big(n_{j,r}, j, st_n^{out}(S_{out}), r\big)$
            count++
        else
            $S_{out} \leftarrow S_{out} \backslash \big(n_{j,r}, j, st_n^{out}(S_{out}), r\big) \bigcup \big(n_{j,r}, j, st_n^{out}(S_{out}), r - \text{count}\big)$
        end if
    end for
end for
for $r = 1; r \leq |N|; r++$ do
    if $status(S_{out}, n_r) = 0$ then
        $j^* \leftarrow \arg\min_{j \in J}\{\ EST_{n_r,j}^{out}(S_{in}, S_{out})\}$
        $S_{out} \leftarrow S_{out} \bigcup \big(n_r, j^*, EST_{n_r,j^*}^{out}(S_{in}, S_{out}), R_{j^*}^{out}(S_{out})\big)$
        $S_{in}, S_{out} \leftarrow VND_4(S_{in}, S_{out})$
    end if
end for
$S_{in}, S_{out} \leftarrow VND_5(S_{in}, S_{out})$
Return $S_{in}, S_{out}$;
```

Algorithm 10 AcceptanceCriterion

```
Inputs : $S^*$, $S'$, $\lambda$;
if $\lambda * f(S') < f(S^*)$ then
    Return $S'$;
else
    Return $S^*$;
end if
```

library of CPLEX 12.6.3. For CPLEX, we allow the solver to use four threads, and we limit the running time to one day (24-hours). For metaheuristics, we run the algorithms 5 times to report the best upper bound, average upper bound, and the worst upper bound generated by the algorithms. The computational experiments we structured as follows. First, we present the data generation method. After that, using the generated data, we analyze the performance of CPLEX solver using the two formulations presented in Section 3.2. Next, we analyze the performance of the branch and cut algorithm. After that, we evaluate the dispatching rules and the composite dispatching rule presented in Section 4.1.1. The next part of the experiments discusses the performance of the heuristics presented in Section 4.1.4. Then, we test the performance of the metaheuristics by comparing their output with the solution generated by the other methods. Finally, we present the sensitivity analysis.

To generate the flow matrix, We applied the method proposed in Guignard et al. (2012) and **?**. The authors assumed that each outbound truck receives commodities from at least one origin, and each inbound truck sends commodities to at least one destination. In addition to that, we assume that the number of commodities to be sent is a random number uniformly distributed between $[1, 5]$. It is assumed that the number of inbound trucks equals the number of outbound trucks and the number of inbound doors equals the number of outbound doors. We generate four sets of data with different flow matrix densities: 25%, 35%, 50%, and 75%. We assume that each commodity takes one unit processing time for both loading and unloading. So, the unloading time of the inbound truck m is $p_m = \sum_{n \in N} w_{(m,n)}$ and loading time of the outbound truck n is $p_n = \sum_{m \in M} w_{(m,n)}$. To generate the distances matrix, we cluster the terminal. Lets, $CL = \max\Big\{2, \min\{\Big\lfloor\frac{|I|}{3}\Big\rfloor, 4\}\Big\}$ be the maximum distance between clusters. Then, the distance between the inbound door i and the outbound door j is $d_{i,j} = \max\{1, \lceil\frac{(i-j)\times CL}{|I|}\rceil, \lceil\frac{(j-i)\times CL}{|I|}\rceil\}$. Note that $d_{i,j} \in \{1, 2, 3, 4\}$. We generate 44 instances and refer to them as 00x00x00. The first two digits correspond to the number of the inbound and outbound trucks. The second two digits corresponds to the number of inbound and outbound doors. And, the last two digits correspond to the density of the flow matrix.

In this part of the experiments, we analyze the performance of the two formulations presented in Section 3.2. We classified the generated instance into four classes based on the flow matrix density. Tables 4.1, 4.2, 4.3, and 4.4 summarizes the results of this experiment. The first column of the table ”instance” contains the name of the instances. The second column ”Opt” contains the best upper bounds obtained by CPLEX after solving both formulations. For each formulation, the tables summarize four information, the LP gap, the final gap, the number of explored nodes, and the total CPU time in seconds. The LP gap is calculated as $100 \times (\text{Opt} - \text{LP})/\text{Opt}$, where LP is the optimal LP relaxation of the problem. The final Gap is obtained by $100 \times (\text{UB} - \text{LB})/\text{UB}$, where UB is the best upper bound and LB is the lower bound obtained by the solver. The ”Nodes” column contains the number of nodes explored in the enumeration tree by the solver. The symbol ”-” means that the solver could not obtain any value.

The results show that there is no clear evidence that one formulation is superior to the other in terms of the LP bound. In fact, for more than a third of the instances, both formulations provide the same LP bound. However, for eight instances Formulation 1 provides better bounds whereas in 15 instances Formulation 2 provides a better LP bound. In average, the LP bound from Formulation 2 is slightly better than the lower bound generated by Formulation 1. Out of 44 instances, CPLEX can solve to optimally 17 instances using Formulation 1 and 18 instances using Formulation 2. For the unsolved instances, the majority remains with a gap of more than 10% at most of 38%. In term of average final gaps and average CPU time, the solver can provide a slightly better final gap in slightly less average time using Formulation 2 as compare to Formulation 1. One significant advantage of Formulation 1 is that Using Formulation 1 the solver can handle larger instances as compared when using Formulation 2. In other words, Formulation 2 needs more RAM due to the large number of constraints (13). For example, the solver can explore 66134 nodes in the enumeration tree while solving 20x10x50 using Formulation 1. On the other hand, the solver got stuck at the root node while solving Formulation 2. To conclude, there is no superior advantage one formulation over the other, but CPLEX can handle larger instance while using Formulation 1.

Table 4.1: Results for instances with 25% density in flow matrix

		Formulation 1 (MIP)				Formulation 2 (IP)			
instance	Opt	LP gap	Final gap	Nodes	CPU time	LP gap	Final gap	Nodes	CPU time
8x4x25	27	0.00	0.00	449	1	0.00	0.00	147	3
9x4x25	33	0.00	0.00	0	1	0.00	0.00	0	1
10x4x25	34	0.00	0.00	0	1	0.00	0.00	0	3
10x5x25	35	0.00	0.00	0	0	0.00	0.00	0	4
11x5x25	36	13.89	0.00	774	13	13.89	0.00	2650	34
12x5x25	46	17.39	0.00	501862	9571	17.17	0.00	1220426	20012
12x6x25	41	0.00	0.00	1206	6	0.00	0.00	76	37
15x6x25	56	21.79	14.29	1999214	86400	20.71	14.29	2810805	86400
15x7x25	46	0.00	0.00	3066799	77796	0.00	0.00	637	182
20x10x25	64	6.25	6.25	1444635	86400	6.25	6.25	542349	86400
50x30x25	-	-	-	0	86400	-	-	0	86400
Average		5.93	2.05		26018.90	5.80	2.05		19307.60

Table 4.2: Results for instances with 35% density in flow matrix

		Formulation 1 (MIP)				Formulation 2 (IP)			
instance	Opt	LP gap	Final gap	Nodes	CPU time	LP gap	Final gap	Nodes	CPU time
8x4x35	43	11.40	0.00	491	4	10.93	0.00	411	9
9x4x35	48	0.00	0.00	159	1	0.00	0.00	239	17
10x4x35	51	15.69	0.00	24802	193	15.69	0.00	58767	1356
10x5x35	44	9.09	0.00	267	8	9.09	0.00	366	37
11x5x35	53	5.66	0.00	644274	22238	5.66	0.00	560073	4879
12x5x35	63	23.02	13.33	2052136	86400	20.32	12.70	982587	86400
12x6x35	52	17.31	13.21	3755253	86400	17.31	13.46	590729	86400
15x6x35	85	10.59	5.88	2321786	86400	10.59	5.88	3294648	86400
15x7x35	71	15.49	15.28	3407403	86400	15.49	8.45	2507594	86400
20x10x35	90	12.22	0.00	172276	19111	10.67	0.00	2	71013
50x30x35	-	-	-	0	86400	-	-	0	86400
Average		12.05	4.77		38715.50	11.57	4.05		42291.10

Table 4.3: Results for instances with 50% density in flow matrix

		Formulation 1 (MIP)				Formulation 2 (IP)			
instance	Opt	LP gap	Final gap	Nodes	CPU time	LP gap	Final gap	Nodes	CPU time
8x4x50	59	18.47	0.00	1873506	21242	20.34	0.00	88946	298
9x4x50	54	16.67	0.00	354303	6552	16.67	0.00	490737	5583
10x4x50	76	22.37	13.82	5727028	86400	25.00	16.71	1427709	86403
10x5x50	69	15.94	0.00	489	20	12.90	0.00	0	98
11x5x50	74	25.68	18.67	5033189	86400	19.73	10.81	4819228	86400
12x5x50	87	27.59	20.80	5773004	86400	23.56	17.05	632744	86400
12x6x50	76	21.71	16.46	6357976	86400	22.37	14.47	4275359	86400
15x6x50	116	34.57	25.86	3509907	86400	30.60	24.14	128402	86400
15x7x50	95	18.72	14.74	2684076	86400	21.28	18.95	385519	86400
20x10x50	120	19.17	15.75	66134	86400	15.17	8.33	0	86400
50x30x50	-	-	-	0	86400	-	-	0	86400
Average		22.09	12.61		63261.40	20.76	11.05		61078.20

Table 4.4: Results for instances with 75% density in flow matrix

		Formulation 1 (MIP)				Formulation 2 (IP)			
instance	Opt	LP gap	Final gap	Nodes	CPU time	LP gap	Final gap	Nodes	CPU time
8x4x75	75	30.67	8.00	5798949	86400	24.53	0.00	2772142	28755
9x4x75	95	32.32	18.95	9688112	86400	32.84	23.96	688577	86402
10x4x75	119	36.95	29.41	7264828	86400	34.49	24.37	626520	86403
10x5x75	95	29.26	22.92	4761122	86400	29.68	13.68	7865677	86400
11x5x75	105	33.81	25.71	4337157	86400	32.86	20.75	1420341	86400
12x5x75	134	29.54	26.67	3640776	86400	34.23	28.36	71930	86400
12x6x75	116	25.86	14.66	6085621	86400	28.45	23.28	555489	86400
15x6x75	179	39.89	33.52	1962730	86400	37.60	38.78	5	86400
15x7x75	159	33.94	28.30	2414206	86400	32.26	35.91	10	86400
20x10x75	211	30.81	30.33	12390	86400	-	-	0	86400
50x30x75	-	-	-	0	86400	-	-	0	86400
Average		32.30	23.85		86400.00	31.88	23.23		80636.00

Table 4.5, summarize the results of branch and cut algorithm (B&C) presented in Section 3.1. We compare its results with the average results obtained from solving Formulation 1 and Formulation 1. For that, we report the final gap, the number of nodes explored, and the CPU time. For the majority of instances, B&C algorithm leads to a weaker Final gap and more computational time. From this experiment, we conclude that adding constraints 13 to Formulation 1 while exploring the enumeration tree does not improve the performance of the commercial solver.

Table 4.6 summarizes the results for each constrictive algorithm and dispatching rule, where we report the maximum, average, and minimum percentage of deviation which we compute as $\%d = 100 \times (\text{Opt} - \text{UB})/\text{Opt}$. Moreover, UB is the upper bound obtained by the algorithm and Opt is the best solution reported obtained throughout the experiments, $\%d_{max}$, $\%d_{avr}$, and $\%d_{min}$ are the maximum, average, and minimum percentage deviation, respectively, obtained for a given algorithm and dispatching rule. To obtain the ranking index value for the composted dispatching rule CDR we set $A = \{P, |L|, TP\}$, $K_P = 0.25$, $K_{|L|} = 10$, and $K_{TP} = 0.35$, for the SSC. Also, we set $A = \{P, |L|, TP\}$, $K_P = 0.25$,

Table 4.5: Results of B&C Algorithm.

	B&C			Average Formulation-(1,2)		
instance	Final gap	Nodes	CPU time	gap	Nodes	CPU time
8x4x25	0.00	105	1	0.00	298	2
8x4x35	0.00	492	5	0.00	451	6.5
8x4x50	0.00	1151473	6412	0.00	981226	10770
8x4x75	19.04	5101464	86402	4.00	4285546	57577.5
9x4x25	0.00	0	7	0.00	0	1
9x4x35	0.00	86	3	0.00	199	9
9x4x50	0.00	271466	5689	0.00	422520	6067.5
9x4x75	26.04	918894	86400	21.45	5188345	86401
10x4x25	0.00	3	1	0.00	0	2
10x4x35	0.00	4180	73	0.00	41784.5	774.5
10x4x50	13.55	1886701	86400	15.26	3577369	86401.5
10x4x75	29.41	757021	86400	26.89	3945674	86401.5
10x5x25	0.00	0	1	0.00	0	2
10x5x35	0.00	660	23	0.00	316.5	22.5
10x5x50	0.00	0	22	0.00	244.5	59
10x5x75	22.11	623356	86400	18.30	6313400	86400.5
11x5x25	0.00	2671	81	0.00	1712	23.5
11x5x35	0.00	421730	6635	0.00	602173.5	13558.5
11x5x50	17.57	2086236	86400	14.74	4926209	86400
11x5x75	25.71	475421	86400	23.23	2878749	86400
12x5x25	2.17	5047509	86400	0.00	861144	14791.5
12x5x35	14.29	1489298	86400	13.02	1517362	86400
12x5x50	20.69	1627756	86400	18.93	3202874	86400
12x5x75	25.00	85498	86400	27.51	1856353	86400
12x6x25	0.00	34	8	0.00	641	21.5
12x6x35	15.38	2644226	86400	13.33	2172991	86400
12x6x50	16.88	1372664	86400	15.46	5316668	86400
12x6x75	17.24	461151	86400	18.97	3320555	86400
15x6x25	14.29	1230570	86400	14.29	2405010	86400
15x6x35	4.71	860961	86400	5.88	2808217	86400
15x6x50	27.97	59095	86400	25.00	1819155	86400
15x6x75	41.58	16256	86400	36.15	981367.5	86400
15x7x25	0.00	3615974	78952	0.00	1533718	38989
15x7x35	16.67	1700115	86400	11.86	2957499	86400
15x7x50	14.74	124914	86400	16.84	1534798	86400
15x7x75	31.10	21561	86400	32.11	1207108	86400
20x10x25	7.69	890111	86400	6.25	993492	86400
20x10x35	0.00	3902	4677	0.00	86139	45062
20x10x50	25.55	7633	86400	12.04	33067	86400
20x10x75	32.73	3851	86400	30.33	12390	86400
50x30x25	-	-	86400	-	-	86400
50x30x35	-	-	86400	-	-	86400
50x30x50	-	-	86400	-	-	86400
50x30x75	-	-	86400	-	-	86400
Average	12.05		57313.45	10.55		55321.45

$K_{|L|} = 10$, and $K_{TP} = 0.4$ for the ISC algorithm. It is clear from the results that ordering trucks in nonincreasing order dominates its inverse. Also, the composite dispatching rule outperforms all other rules and LSP comes in the second place.

Table 4.6: Results of constructive algorithms and dispatching rules

		LSP	SPT	MNLT	LNLT	BTAL	STAL	CDR
SSC	$\%d_{max}$	29.63	48.24	22.22	50.94	30.77	50.59	18.52
	$\%d_{avr}$	11.03	30.56	13.37	28.44	14.08	27.19	10.19
	$\%d_{min}$	0.00	11.68	5.22	12.00	3.33	12.63	0.00
ISC	$\%d_{max}$	24.53	33.33	35.29	35.85	29.63	40.67	24.07
	$\%d_{avr}$	12.09	20.70	16.48	20.11	15.67	21.61	11.56
	$\%d_{min}$	3.05	9.47	5.08	6.29	8.12	5.33	2.00

Table 4.7 summarizes the results for SSCLS and ISCLS algorithms using two dispatching rules: SPT and CDR. The header line is the dispatching rule used by the algorithm. For each rule, the table demonstrates the average percentage deviation and the average running time (in seconds). Moreover, for each algorithm, the table shows the average and the maximum values obtained after solving all the instances. For the composite dispatching rule, we used the same setting used in the previous experiment. The results show that SSCLS performs better than ISCLS in terms of both percentage of deviation and running time. Also, the composite dispatching rule outperforms the largest process time first rule. Based on these results, we used the composite dispatching rule to order trucks in metaheuristics.

Table 4.7: Results of composite heuristics algorithms

		LSP		CDR	
instance		d%	Time	d%	Time
SSCLS	Max	8.33	106.00	6.76	170.00
	Avr	2.60	4.95	2.45	7.66
ISCLS	Max	9.09	172.00	9.26	175.00
	Avr	3.64	7.16	3.37	7.59

Table 4.8 summarizes the results from the GRASP and ILS metaheuristics algorithm. For each instance, we run each algorithm 50 times and then we report the average, maximum, and minimum percentage deviation. In both metaheuristics, trucks are ordered based

on the composite dispatching rule using equations (17) and (18). To control the ILS algorithm we set $p_o = 0.05$, $p_{max} = 0.30$, and $\delta = 0.05$. On the other hand, to control the GRASP, we set $\alpha_o = 0.7$, α_{min}, and $\delta = 0.95$. The results show that both metaheuristics provide high-quality solutions in a reasonable amount of time. In average ILS obtains upper bounds that are 1.04 % away from the best-known solution. On the other hand, GRASP is away by 1.58%. Also, in terms of maximum deviation, both metaheuristics deviate at most by 6.25% of the best known upper bound. In all criteria, ILS performs better than GRASP. ILS performs better than GRASP on the majority of the considered instances.

Table 4.9 provides an overall comparison and summarizes the results of SSC, SSCLS, ILS, GRASP, and results obtained by the solver. Our objective is to analyze and compare the quality of the solution generated by each method and the computational time. We report the best results from the solver using both formulations. The table shows that both metaheuristics are able to obtain better solutions than CPLEX for 11 instances. SSC provides an upper in less than a second for all instances with an average gap of 10.2 % and a maximum gap of 18.5 %. Also, experiments show that SSC performance is related to the size of the instance and the ratio of the number of trucks to the number of doors. Instances with more trucks tend to perform better in term of percentage deviations. On the other hand, worst performance comes from the instances with the smallest number of trucks. Also, better performance is expected when we have lower ratio of the trucks to doors.

SSCLS obtains better solution that SSC. By exploring the neighborhoods, the algorithm can reduce the gap by up to 11.76 %. The SSCLS heuristics obtain better solutions on the cost of increasing the computational time. For all instances with less than 50 trucks, the computational time are less than two seconds, whereas, for instances with 50 trucks computational time is more than 29 seconds. The of the quality of solutions obtain from the metaheuristics are better than the SSC and SSCLS heuristics in term of percentage of deviation. However, metaheuristics need more computational effort for obtaining these

Table 4.8: Results of metaheuristics algorithm.

		ILS				GRASP			
instance	Opt	$\%d_{Avr}$	$\%d_{Max}$	$\%d_{Min}$	CPU	$\%d_{Avr}$	$\%d_{Max}$	$\%d_{Min}$	CPU
8x4x25	27	2.44	3.70	0.00	0.04	3.04	3.70	0.00	0.06
8x4x35	43	0.23	2.33	0.00	0.04	0.42	2.33	0.00	0.06
8x4x50	59	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.08
8x4x75	75	0.00	0.00	0.00	0.08	0.03	1.33	0.00	0.06
9x4x25	33	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.1
9x4x35	48	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.1
9x4x50	54	0.85	3.70	0.00	0.1	4.48	5.56	0.00	0.12
9x4x75	95	0.02	1.05	0.00	0.14	1.60	3.16	0.00	0.14
10x4x25	34	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.12
10x4x35	51	0.35	1.96	0.00	0.14	3.06	3.92	0.00	0.14
10x4x50	76	0.95	2.63	0.00	0.16	2.37	3.95	0.00	0.18
10x4x75	117	0.94	1.71	0.00	0.26	1.23	1.71	0.00	0.28
10x5x25	35	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.1
10x5x35	44	2.05	4.55	0.00	0.1	4.23	4.55	0.00	0.14
10x5x50	69	0.00	0.00	0.00	0.16	0.43	1.45	0.00	0.18
10x5x75	95	0.00	0.00	0.00	0.2	0.00	0.00	0.00	0.22
11x5x25	36	2.11	2.78	0.00	0.16	2.50	2.78	0.00	0.18
11x5x35	53	0.00	0.00	0.00	0.16	0.00	0.00	0.00	0.22
11x5x50	74	2.00	4.05	0.00	0.26	3.03	5.41	0.00	0.26
11x5x75	105	1.28	1.90	0.00	0.34	1.77	2.86	0.95	0.38
12x5x25	46	0.00	0.00	0.00	0.22	0.13	2.17	0.00	0.28
12x5x35	63	1.59	3.17	0.00	0.26	3.21	4.76	1.59	0.3
12x5x50	87	1.79	3.45	1.15	0.38	2.41	3.45	1.15	0.42
12x5x75	130	1.55	3.08	0.00	0.54	2.51	3.85	0.00	0.54
12x6x25	41	0.00	0.00	0.00	0.18	1.95	2.44	0.00	0.24
12x6x35	52	1.50	3.85	0.00	0.26	1.65	1.92	0.00	0.3
12x6x50	76	2.29	3.95	0.00	0.36	2.66	3.95	1.32	0.38
12x6x75	115	0.82	0.87	0.00	0.46	0.87	0.87	0.87	0.48
15x6x25	56	2.07	3.57	0.00	0.56	2.89	3.57	0.00	0.68
15x6x35	85	1.11	2.35	0.00	0.72	1.34	2.35	0.00	0.82
15x6x50	115	1.46	2.61	0.00	1.02	1.51	2.61	0.00	1.06
15x6x75	175	1.47	2.86	0.57	1.44	1.45	2.29	0.00	1.5
15x7x25	46	2.35	4.35	0.00	0.5	2.30	4.35	2.17	0.66
15x7x35	71	2.34	2.82	1.41	0.7	2.45	2.82	0.00	0.78
15x7x50	94	1.28	2.13	0.00	0.9	2.00	3.19	1.06	0.96
15x7x75	156	0.64	1.28	0.00	1.38	0.69	1.28	0.64	1.38
20x10x25	64	3.56	6.25	1.56	1.72	4.19	6.25	1.56	2.08
20x10x35	90	2.29	3.33	0.00	2.3	2.44	3.33	1.11	2.64
20x10x50	119	2.18	3.36	0.84	3.24	1.92	2.52	0.84	3.44
20x10x75	197	0.68	1.02	0.00	4.6	0.84	1.02	0.51	4.9
50x30x25	150	0.00	0.00	0.00	117.14	0.00	0.00	0.00	127.68
50x30x35	199	0.70	1.01	0.00	169.36	0.90	1.51	0.00	176.82
50x30x50	291	0.67	1.03	0.34	223.8	0.72	1.03	0.00	227.98
50x30x75	441	0.20	0.45	0.00	297.38	0.29	0.45	0.00	310.96
Maximum	441.00	3.56	6.25	1.56	297.38	4.48	6.25	2.17	310.96
Average	95.05	1.04	1.98	0.13	18.91	1.58	2.38	0.31	19.78

solutions. The solver obtained the optimal solution for 18 instances. Out of that, ILS obtained the optimal solution for 10 instances and GRASP obtained the optimal solution for eight instances. In general, ILS performs better than GRASP. In term of average, ILS provides a better solution than GRASP in less amount of time.

For testing assumptions of the model, we conduct two experiments. In the first one, we analyze the importance of including internal transportation time in the model. In the second one, we analyze the benefits of synchronizing the truck assignment decision with the truck scheduling decision. The first test tries to answer the question, should we include the internal transportation time or should we neglect it? To answer this question, we present three cases. In the first case, we assume that the decision maker neglects the internal transportation time at the cross-dock terminal. In the second, we assume that he includes the transshipments time. Finally, we assume the decision maker overestimate the internal transportation time.

Tables 4.10, 4.11, and 4.12 summarize the analysis of the internal transportation assumption. The first column is the door index. The second column contains the inbound doors schedule followed by the outbound doors schedule. Let (m, t) denotes the truck schedule, where m is the truck index and t the starting time of the truck m. $\bar{C}_{max}$ denotes makespan resulted of a given case. C_{max} denotes the actual makespan that is resulted optimal solution of our model. The last column contains the percentage deviation obtained for each policy. The first case assumes that the distance between inbound and outbound doors are zero. The second case corresponds to the original problem. The final case doubles the distance between between inbound door and outbound door. The results show that in both cases neglecting and over estimating the internal transportation time between time results in change of outputs. Also, in the case of neglecting the transpiration time, trucks scheduling and assignment are entirely changed. Also, this case leads to 8.3$\%$ deviation from optimal solution. In the other hand, overestimating the internal transportation costs

Table 4.9: Overall Comparison

		Const-1	COMP H1		ILS		GRASP		CPLEX Best	
instance	Opt	$\%d$	$\%d$	Time	$\%d_{avr}$	avrTime	$\%d_{avr}$	avrTime	$\%d_{avr}$	avrTime
8x4x25	27	18.52	3.70	0.00	2.44	0.04	3.04	0.06	0.00	1.00
8x4x35	43	11.63	2.33	0.00	0.23	0.04	0.42	0.06	0.00	4.00
8x4x50	59	5.08	1.69	0.00	0.00	0.06	0.00	0.08	0.00	6412.00
8x4x75	75	1.33	0.00	0.00	0.00	0.08	0.03	0.06	0.00	86400.00
9x4x25	33	9.09	0.00	0.00	0.00	0.06	0.00	0.10	0.00	1.00
9x4x35	48	14.58	2.08	0.00	0.00	0.08	0.00	0.10	0.00	1.00
9x4x50	54	12.96	5.56	0.00	0.85	0.10	4.48	0.12	0.00	5689.00
9x4x75	95	12.63	2.11	0.00	0.02	0.14	1.60	0.14	0.00	86400.00
10x4x25	34	5.88	0.00	0.00	0.00	0.08	0.00	0.12	0.00	1.00
10x4x35	51	13.73	3.92	0.00	0.35	0.14	3.06	0.14	0.00	73.00
10x4x50	76	14.47	2.63	0.00	0.95	0.16	2.37	0.18	0.00	86400.00
10x4x75	117	13.68	3.42	0.00	0.94	0.26	1.23	0.28	1.71	86400.00
10x5x25	35	2.86	0.00	0.00	0.00	0.08	0.00	0.10	0.00	0.00
10x5x35	44	15.91	4.55	0.00	2.05	0.10	4.23	0.14	0.00	8.00
10x5x50	69	8.70	0.00	0.00	0.00	0.16	0.43	0.18	0.00	20.00
10x5x75	95	3.16	1.05	0.00	0.00	0.20	0.00	0.22	0.00	86400.00
11x5x25	36	16.67	5.56	0.00	2.11	0.16	2.50	0.18	0.00	13.00
11x5x35	53	11.32	0.00	0.00	0.00	0.16	0.00	0.22	0.00	6635.00
11x5x50	74	13.51	6.76	0.00	2.00	0.26	3.03	0.26	0.00	86400.00
11x5x75	105	18.10	6.67	0.00	1.28	0.34	1.77	0.38	0.00	86400.00
12x5x25	46	6.52	4.35	0.00	0.00	0.22	0.13	0.28	0.00	9571.00
12x5x35	63	11.11	3.17	0.00	1.59	0.26	3.21	0.30	0.00	86400.00
12x5x50	87	14.94	3.45	0.00	1.79	0.38	2.41	0.42	0.00	86400.00
12x5x75	130	16.92	2.31	0.00	1.55	0.54	2.51	0.54	1.54	86400.00
12x6x25	41	12.20	0.00	0.00	0.00	0.18	1.95	0.24	0.00	6.00
12x6x35	52	11.54	3.85	0.00	1.50	0.26	1.65	0.30	0.00	86400.00
12x6x50	76	9.21	3.95	0.00	2.29	0.36	2.66	0.38	0.00	86400.00
12x6x75	115	5.22	1.74	0.00	0.82	0.46	0.87	0.48	0.87	86400.00
15x6x25	56	12.50	3.57	0.00	2.07	0.56	2.89	0.68	0.00	86400.00
15x6x35	85	16.47	2.35	0.00	1.11	0.72	1.34	0.82	0.00	86400.00
15x6x50	115	11.30	1.74	0.00	1.46	1.02	1.51	1.06	0.87	86400.00
15x6x75	175	13.14	1.71	1.00	1.47	1.44	1.45	1.50	2.29	86400.00
15x7x25	46	15.22	0.00	0.00	2.35	0.50	2.30	0.66	0.00	77796.00
15x7x35	71	9.86	2.82	0.00	2.34	0.70	2.45	0.78	0.00	86400.00
15x7x50	94	7.45	3.19	0.00	1.28	0.90	2.00	0.96	1.06	86400.00
15x7x75	156	13.46	3.21	0.00	0.64	1.38	0.69	1.38	1.92	86400.00
20x10x25	64	10.94	4.69	0.00	3.56	1.72	4.19	2.08	0.00	86400.00
20x10x35	90	6.67	2.22	1.00	2.29	2.30	2.44	2.64	0.00	4677.00
20x10x50	119	9.24	1.68	0.00	2.18	3.24	1.92	3.44	0.84	86400.00
20x10x75	197	3.05	2.03	1.00	0.68	4.60	0.84	4.90	7.11	86400.00
50x30x25	150	0.00	0.00	29.00	0.00	117.14	0.00	127.68	-	86400.00
50x30x35	199	3.52	1.51	48.00	0.70	169.36	0.90	176.82	-	86400.00
50x30x50	291	2.06	1.37	87.00	0.67	223.80	0.72	227.98	-	86400.00
50x30x75	441	1.81	0.91	170.00	0.20	297.38	0.29	310.96	-	86400.00
Maximum	441.00	18.52	6.76	170.00	3.56	297.38	4.48	310.96	-	86400.00
Average	95.05	10.19	2.45	7.66	1.04	18.91	1.58	19.78	-	55538.82

Table 4.10: Results of case 1: neglecting the internal transportation time

door	inbound Schedule	outbound Schedule	$\bar{C}_{max}$	C_{max}	$\%d$
0	(8 0), (3 12)	(1 12), (8 19), (2 24)			
1	(10 0), (7 12)	(10 17), (4 24)			
2	(5 0), (0 13), (1 20)	(5 13), (9 25)	33	39	8.3%
3	(9 0), (2 6)	(7 12), (0 20)			
4	(4 4), (6 8)	(3 17), (6 26)			

Table 4.11: Results of case 2: including the transshipment time

door	Inbound Schedule	outbound Schedule	$\bar{C}_{max}$	C_{max}	$\%d$
0	(2 0), (7 11)	(1 20), (2 27)			
1	(10 0), (9 12)	(0 15), (4 27)			
2	(3 0), (8 3)	(10 14), (7 21), (6 29)	36	36	0.0%
3	(6 0), (4 9), (1 19)	(3 18), (9 27)			
4	(5 0), (0 17)	(5 18), (8 30)			

Table 4.12: Results of case 3: over estimating the transshipment time

door	inbound schedule	outbound schedule	$\bar{C}_{max}$	C_{max}	$\%d$
0	(8 0), (9 12), (1 19)	(1 22), (4 29)			
1	(6 0), (7 9)	(3 20), (2 30)			
2	(2 0), (0 11)	(10 17), (7 24), (6 32)	39	37	2.8%
3	(3 0), (5 3)	(0 19), (9 31)			
4	(10 0), (4 12)	(8 21), (5 26)			

results in 2.5% deviation from the optimal solution.

The last experiment analyzes the effects of synchronizing the assignment and scheduling decisions. It answers the question, is it beneficial to synchronize the assignment and scheduling decisions? To answer this question, we solve a desynchronize model and compare the result of that with the results of our model. In the desynchronize model, we assume that the decision maker first assigns trucks to doors and then schedules trucks at each door. In the first stage, the objective is to minimize internal transportation time. In the second stage, the objective is to minimize the makespan and we assume that doors have capacities.

Let $\bar{p}$ denotes the average process time. We consider three scenarios each with a different capacity, In scenario one, we set the capacity to $cap = 2.5 * \bar{p}$. In scenario two, we set $cap = 3 * \bar{p}$. Finally, in scenario three, we set $cap = 4 * \bar{p}$. The results of this experiment are summarised in table 4.13, The result shows that synchronized leads to better decisions compared the desynchronized mode.

Table 4.13: Results of desynchronize model

Capacity	door	inbound schedule	outbound schedule	$\bar{C}_{max}$	$\%d$
$2.5\bar{p}$	0	(10 0), (3 12)	(7 17), (9 28)	38	5.6%
	1	(9 0), (5 6)	(10 18), (0 25)		
	2	(7 0), (0 12)	(1 21), (3 28)		
	3	(2 0), (6 11)	(2 16), (5 24)		
	4	(1 0), (8 4), (4 16)	(4 16), (6 24), (8 30)		
$3\bar{p}$	0	(6 0), (1 9)		43	19.4%
	1	(5 0)	(3 19), (6 28), (4 34)		
	2	(8 0), (7 12)	(0 18), (2 34)		
	3	(2 0), (3 11), (4 14), (9 22)	(8 14), (5 23), (9 35)		
	4	(0 0), (10 3)	(10 20), (1 27), (7 34)		
$4\bar{p}$	0	(5 0), (2 13), (1 24)	(8 16), (1 26), (0 33), (2 45)	54	50.0%
	1	(4 0), (0 4), (7 7), (6 19), (3 29)	(5 20), (4 34), (3 42)		
	2	(8 0), (10 12), (9 24)	(9 25), (6 32), (7 38), (10 46)		
	3				
	4				

Chapter 5

Conclusion

In this thesis, we studied the truck to door scheduling problem. The problem is to assign and schedule trucks to doors in a way that minimizes the makespan. We considered that the internal transportation time inside the terminal in an essential factor for making these decision. To obtain a better decision, truck to door assignments and truck scheduling decisions are made simultaneously. We presented several dispatching rules and presented a composite dispatching rule. We used these dispatching rules to develop several heuristics and metaheuristics. We presented constructive heuristics, local search procedures, compound constructive local search heuristics algorithms, and two metaheuristics algorithms.

We presented and solved two mathematical formulations using the commercial software CPLEX and presented computational experiments. We found that neither formulation dominates the other. Also, we observed that the complexity of the problem is directly related to the density of the flow matrix between in inbound trucks and outbound trucks. For the 75% flow, CPLEX was not able to solve most of the considered instances. Also, for the largest instance with 50 inbound trucks and 50 outbound trucks, we run out of memory before the solver started solving the problem. We found that heuristics algorithms were successful in finding a good quality solution. Also, the metaheuristic provides near to optimal solutions and in some cases the optimal solution. For some cases where CPLEX could not find the optimal solution, the metaheuristics algorithm was able to generate better upper bounds. Among the two metaheuristics, we found that the Iterated Local Search provides better solutions.

For future research, several directions are possible. More work could be done on the solution methodology for both exact and heuristic algorithms. One could use the structure of the problem to decompose the problem into several problems and then implement a hybrid heuristic or metaheuristic. Also, several objective functions could be adapted to the model such as the number of tardy trucks, total weighted earliness, and lateness of the trucks, and total completion time.

Bibliography

Uday M Apte and S Viswanathan. Effective cross docking for improving distribution efficiencies. *International Journal of Logistics*, 3(3):291–302, 2000.

AR Boloori Arabani, SMT Fatemi Ghomi, and Mostafa Zandieh. Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert systems with Applications*, 38(3):1964–1979, 2011.

Mohammad Taghi Assadi and Mohsen Bagheri. Differential evolution and population-based simulated annealing for truck scheduling problem in multiple door cross-docking systems. *Computers & Industrial Engineering*, 96:149–161, 2016.

Adrien Bellanger, SaïD Hanafi, and Christophe Wilbaut. Three-stage hybrid-flowshop model for cross-docking. *Computers & Operations Research*, 40(4):1109–1121, 2013.

Peter Bodnar, René de Koster, and Kaveh Azadeh. Scheduling trucks in a cross-dock with mixed service mode dock doors. *Transportation Science*, 51(1):112–131, 2015.

Nils Boysen. Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research*, 37(1):32–41, 2010.

Nils Boysen and Malte Fliedner. Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413–422, 2010.

Nils Boysen, Malte Fliedner, and Armin Scholl. Scheduling inbound and outbound trucks at cross docking terminals. *OR spectrum*, 32(1):135–161, 2010.

Paul Buijs, Iris FA Vis, and Héctor J Carlo. Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 239 (3):593–608, 2014.

Feng Chen and Chung-Yee Lee. Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research*, 193(1):59–72, 2009.

Feng Chen and Kailei Song. Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers & Operations Research*, 36(6):2066–2073, 2009.

Rongjun Chen, Baoqiang Fan, and Guochun Tang. Scheduling problems in cross docking. In *International Conference on Combinatorial Optimization and Applications*, pages 421–429. Springer, 2009.

Priscila M Cota, Bárbara MR Gimenez, Dhiego PM Araújo, Thiago H Nogueira, Mauricio C de Souza, and Martín G Ravetti. Time-indexed formulation and polynomial time heuristic for a multi-dock truck scheduling problem in a cross-docking centre. *Computers & Industrial Engineering*, 95:135–143, 2016.

Paola Festa and Mauricio GC Resende. An annotated bibliography of grasp–part ii: Applications. *International Transactions in Operational Research*, 16(2):131–172, 2009.

Gabriela B Fonseca, Thiago H Nogueira, and Martin G Ravetti. A hybrid lagrangean metaheuristic for the two-machine cross-docking flow shop scheduling problem. *arXiv preprint arXiv:1702.05603*, 2017.

Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

Kevin R Gue. The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*, 33(4):419–428, 1999.

Monique Guignard, Peter M Hahn, A Alves Pessoa, and Daniel Cardoso da Silva. Algorithms for the cross-dock door assignment problem. In *Proceedings of the Fourth International Workshop on Model-Based Metaheuristics*, 2012.

Yu Guo, Zhou-Rong Chen, Yong-Liu Ruan, and Jun Zhang. Application of nsga-ii with local search to multi-dock cross-docking sheduling problem. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 779–784. IEEE, 2012.

Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

Dror Hermel, Hamed Hasheminia, Nicole Adler, and Michael J Fry. A solution framework for the multi-mode resource-constrained cross-dock scheduling problem. *Omega*, 59: 157–170, 2016.

Cheol Min Joo and Byung Soo Kim. Scheduling compound trucks in multi-door cross-docking terminals. *The International Journal of Advanced Manufacturing Technology*, 64(5-8):977–988, 2013.

M Keshtzari, B Naderi, and E Mehdizadeh. An improved mathematical model and a hybrid metaheuristic for truck scheduling in cross-dock problems. *Computers & Industrial Engineering*, 91:197–204, 2016.

Yiyo Kuo. Optimizing truck sequencing and truck dock assignment in a cross docking system. *Expert Systems with Applications*, 40(14):5532–5541, 2013.

Anne-Laure Ladier and Gülgün Alpan. Cross-docking operations: Current research versus industry practice. *Omega*, 62:145–162, 2016.

Yanzhi Li, Andrew Lim, and Brian Rodrigues. Crossdockingjit scheduling with time windows. *Journal of the Operational Research Society*, 55(12):1342–1351, 2004.

Zhaowei Miao, Shun Cai, and Di Xu. Applying an adaptive tabu search algorithm to optimize truck-dock assignment in the crossdock management system. *Expert Systems with Applications*, 41(1):16–22, 2014.

Shrikant S Panwalkar and Wafik Iskander. A survey of scheduling rules. *Operations research*, 25(1):45–61, 1977.

Jacques Roy. Logistics and the competitiveness of canadian supply chains. *Global Value Chains: Impacts and Implications*, 2011.

Mojtaba Shakeri, Malcolm Yoke Hean Low, Stephen John Turner, and Eng Wah Lee. A robust two-phase heuristic algorithm for the truck scheduling problem in a resource-constrained crossdock. *Computers & Operations Research*, 39(11):2564–2577, 2012.

Mojtaba Shakeri, Malcolm Yoke Hean Low, Stephen John Turner, and Eng Wah Lee. An efficient incremental evaluation function for optimizing truck scheduling in a resource-constrained crossdock using metaheuristics. *Expert Systems with Applications*, 45:172–184, 2016.

André Luís Shiguemoto, Ubiratan Soares Cavalcante Netto, and Gabriela Helena Sergio Bauab. An efficient hybrid meta-heuristic for a cross-docking system with temporary storage. *International Journal of Production Research*, 52(4):1231–1239, 2014.

Kailei Song and Feng Chen. Scheduling cross docking logistics optimization problem with multiple inbound vehicles and one outbound vehicle. In *Automation and Logistics, 2007 IEEE International Conference on*, pages 3089–3094. IEEE, 2007.

Konrad Stephan and Nils Boysen. Cross-docking. *Journal of Management Control*, 22(1): 129, 2011.

Supply Chain Management Association. The next frontier of value creation: The economic and strategic impact of supply chain management in canada, 2016. URL http://scma.com/images/scma/Resources/SCMA-next-frontier-of-value-creation-august-2016.pdf.

Transport Canada. Transportation in canada 2016. Report 978-0-660-08415-2, Transport Canada, 2017. URL https://www.tc.gc.ca/eng/policy/transportation-canada-2016.html.

Behnam Vahdani and Mostafa Zandieh. Scheduling trucks in cross-docking systems: Robust meta-heuristics. *Computers & Industrial Engineering*, 58(1):12–24, 2010.

Jan Van Belle, Paul Valckenaers, and Dirk Cattrysse. Cross-docking: State of the art. *Omega*, 40(6):827–846, 2012.

Jan Van Belle, Paul Valckenaers, Greet Vanden Berghe, and Dirk Cattrysse. A tabu search approach to the truck scheduling problem with multiple docks and time windows. *Computers & Industrial Engineering*, 66(4):818–826, 2013.

Warisa Wisittipanich and Piya Hengmeechai. Truck scheduling in multi-door cross docking terminal by modified particle swarm optimization. *Computers & Industrial Engineering*, 113:793–802, 2017.

Mehdi Yazdani, B Naderi, and M Mousakhani. A model and metaheuristic for truck scheduling in multi-door cross-dock problems. *Intelligent Automation & Soft Computing*, 21(4):633–644, 2015.

Yan Ye, Jingfeng Li, Kaibin Li, and Hui Fu. Cross-docking truck scheduling with product unloading/loading constraints based on an improved particle swarm optimisation algorithm. *International Journal of Production Research*, pages 1–21, 2018.

Wooyeon Yu. Operational strategies for cross docking systems. 2002.

Wooyeon Yu and Pius J Egbelu. Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184(1):377–396, 2008.