Heuristics for Truck Scheduling at Cross Docking Terminals

Wenying Yan

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 in Industrial Engineering

Concordia University

Montreal, Quebec, Canada

March 2014

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Wenying Yan**

Entitled: **"Heuristics for Truck Scheduling at Cross Docking Terminals"**
and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Industrial Engineering

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

Signed by the final examining committee:

Dr. C.Chen Chair

Dr. O. Kuzgunkaya Examiner

Dr. N. Vidyarthi Examiner

Dr. I. Contreras Supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____
Dean of Faculty

April 7, 2014

Abstract

Heuristics for Truck Scheduling at Cross Docking Terminals

Wenying Yan

Cross-docking is a logistics management concept that has been gaining global recognition in less-than-truckload logistics industries and retail firms. In cross-docking terminals, shipments are unloaded from inbound trucks at strip doors, consolidated insider cross-docks according to their destinations, and then, loaded into outbound tucks at stack doors. The goal of cross-docking is to reduce inventory and order picking which are the two most costly functions of traditional warehousing management. The sequence in which the inbound and outbound trucks have to be processed at the cross-dock is crucial for improving the efficiency of cross-docking systems. In this thesis we introduce an integer programming formulation and apply four heuristic algorithms: a local search, a simulated annealing, a large neighborhood search and a beam search, to schedule the trucks in a cross-docking terminal so as to minimize the total operational time.

Acknowledgements

I would like to thank all people who help me to make this work possible. First, and most importantly, I would like to express my sincere gratitude to my supervisor Dr. Ivan Contreras for his support, supervision and encouragement throughout this research. I am very honored to become his student.

I would like to thank all the colleagues in the lab to help me in the thesis. I would like to express my sincere appreciation to my friends, Doc Livingston, Pierre C. Pepin and Diane Pepin to help me settle down in Canada.

我要永远感谢我的外公陈恢美, 外婆: 范清秀, 父母: 严良国和陈明英, 妹妹: 严虹, 弟弟: 严文超以及其他所有的亲戚和朋友在我需要帮助的时候总能伸出援助之手.

Table of Content

List of Figures

List of Tables

Chapter 1: Introduction

The contribution of the logistics service industry to Canada GDP is reported to increase 47% since 1998. Logistics service providers GDP was predicted to continually increase by 40% between 2007 and 2015, generating $56 billion dollars [1]. In 2011, truck transportation shared the largest segment of logistics services and accounted for 31% of the sectors share of GDP; the air and rail segments represented 12% and 11% respectively [2]. These numbers illuminate the importance of logistics and related areas (e.g. supply chain management). However, the synchronization of the distribution of goods with supply chain partners is an extremely complex strategic issue for decision makers. One innovative strategy in logistics and supply chain management that has increasingly attracted industrial practitioners and researchers is cross-docking [3-6].

The main idea of cross-docking is to transfer freights directly from the inbound trucks to the outbound trucks, without or little storage in between. This leads to reduction of the two most costly operations (inventory and order picking) in warehouses. In a traditional warehouse, goods are first received from the suppliers and stored without knowing the demand. When customers order some products, the workers pick them from the pallet racks and send them to the customers. For some products that are expensive to be kept in inventory or that cannot be kept for long time (e.g. perishable food items), the

implementation of a cross-docking strategy exerts enormous advantages [7-10]. Another important goal of cross-docking is to consolidate products from different suppliers to the same destination carried by full loaded trucks, so that economies of scale in transportation costs can be achieved [3].

The appropriate coordination of inbound and outbound trucks plays a crucial role in the efficiency of cross-docking systems as well as that of the total supply chain system. Truck scheduling operations consider the assignment of inbound trucks to receiving (or strip) doors where the freight is unloaded, and the assignment of outbound trucks to shipping (or stack) doors where the freight is loaded. At the beginning of the planning horizon, inbound trucks arrive at the cross-dock. According to the demands of the outbound trucks, a certain sequence is planned in such a way that the makespan is minimized. Products are then unloaded onto receiving docks and moved from strip doors to stack doors by some material handling systems, such as conveyors or forklifts. If the unloaded products are not required to be loaded on outbound trucks at the same time they arrive, the products will be stored temporarily until the outbound trucks, which request of those products, are assigned to the stack doors. Therefore, two interrelated questions are raised by decision makers: which docks the trucks should be assigned to and when their associated products should be loaded or unloaded.

Even though there is an interest for practitioners and researchers in studying optimization

problems arising in cross-docking (see [4, 16]), there are few papers addressing truck scheduling problems. In this thesis, we study a truck scheduling problem recently introduced by Boysen et al. [11]. Due to the considerable complexity of truck scheduling, this problem focuses on cross-docks having a single strip and a single stack door to derive a base model. This is indeed a fundamental problem in cross-docking which helps to gain insight into the underlying structure of the problem and to provide the starting point to more complex settings. Unlike McWilliams et al. [12] and Yu and Egbelu [13], which both study more detailed truck scheduling problems, we deal with the problem on a more aggregate level. Different amounts of goods and the sequence of the inbound trucks lead to the different unload handling times. The loaded handling times for outbound trucks also vary according to the different demands of the customers. What is more, it is already very complex to determinate the transportation times inside the cross-dock from doors to doors. Therefore, the average times are barely useful to estimate the handling times in the detailed truck scheduling model. Under the above analysis, aggregate models outperform detailed models because the latter may result in more confusion and even infeasible solutions. As an aggregate view, the time horizon is distributed to service time slots (represented with unit times such as hours or minutes), on the assumption that trucks can be completely unloaded or loaded within such a time slot. Delivery times from the inbound doors to the outbound doors inside the cross-dock can be defined as a delay (measured by number of slots).

The objective of this research is to find inbound and outbound truck sequences so that the total operational time is minimized. The main contribution of this thesis is to develop four heuristic algorithms to solve the problem: a local search, a simulated annealing, a large neighborhood search, and a beam search. We also propose an integer programming formulation that is different from the one proposed in Boysen et al. [11]. Moreover, motivated by the flow structure of real applications, we introduce two new sets of instances to assess the performance of the proposed solution methods. Computational results obtained on benchmark instances from [11] and two new sets of instances confirm the efficiency of the proposed algorithms.

The remainder of this thesis is organized as follows. Chapter 2 provides a literature review of cross-docking. The formal definition of the studied problem and the new formulation are described in Chapter 3. We then present the four heuristics in Chapter 4. In Chapter 5 we show the computational analysis to compare the efficiency of formulations and algorithms. Finally, in Chapter 6 we summarize our conclusions and point out future research directions.

Chapter 2: Literature Review

In today's globally competitive business environment, more and more innovations focus on the improvement of the whole supply chain perspective as compared to that of a company level. Cross-docking is one of these innovative strategies that help supply chain players synchronize and work together to exert high efficiency and effectiveness. In this chapter we introduce the definitions of cross-docking and present its applications and research trends. We then discuss conditions to properly apply cross-docking. After that, different problems (from a strategical point of view to an operational perspective) raised when designing and implementing cross-docking systems are described. Given that this thesis focuses on truck scheduling, the last section is dedicated to introduce some relevant details related to this topic.

2.1 Cross-docking

There are several definitions of cross-docking. We can summarize cross-docking as the process of receiving products from several suppliers or manufacturers and consolidating those products in a cross-dock terminal to be then delivered to common destinations. By appropriately synchronizing inbound trucking and outbound trucking, the merchandise is able to immediately transfer from the receiving docks to the shipping docks without putting it first in storage location [14, 15]. However, many researchers relax the perfect

synchronization as it is very difficult to achieve. What is more, in practice, before an outbound truck can be assigned, temporary storage is necessary because many receiving shipments from different receiving docks need to be sorted and consolidated. Therefore, cross-docking can also be described as the process of consolidating freight with the same destination (but coming from several origins), with minimal handling and with little or no storage between unloading and loading of the goods [16].

A cross-dock is the terminal dedicated for cross-docking and it consists of several strip doors where inbound trucks are assigned and the freight is unloaded, and several stack doors where outbound trucks are assigned and the freight is loaded. Figure 1 gives a schematic representation of the material handling operations at a long, narrow rectangle-shaped (or I-shaped: the most common shape) cross-dock with 6 strip docks and stack doors.

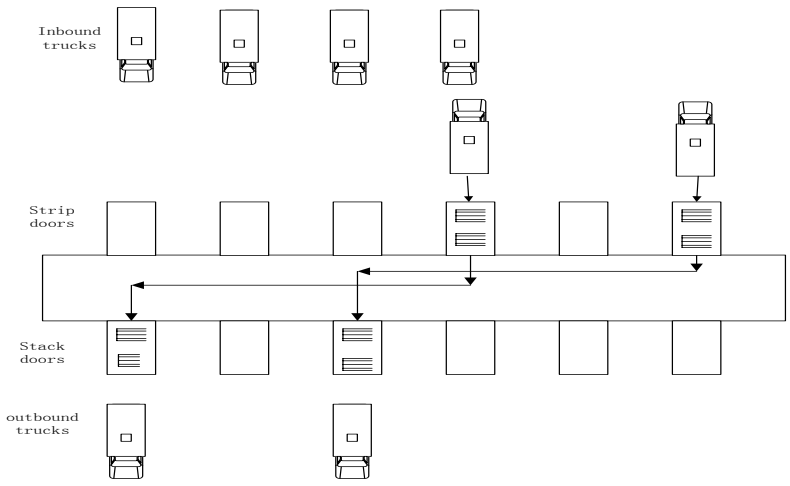

Figure 1. Freight flow in a cross-docking terminal

From the time cross-docking was first used by the US trucking industry during the 1930s [17] until now, it has been successfully implemented globally. With cross-docking on hand, Wal-Mart achieved the goal of delivering the requested goods punctually to customers in different locations [18]. Nowadays, almost all third-party logistics companies in Hong Kong are applying cross-docking systems [19]. In Germany, the travel distance was reduced by 37-39% at a parcel sorting center of Deutsche Post World Net [20] by applying cross-docking. Many other companies have also reported the successful application of cross-docking (e.g. Eastman Kodak Co. [21], Goodyear GB Ltd., Dots, LLC [22] and Toyota [23]). Its successful implementation revealed the advantages of cross-docking as compared to traditional distribution centers: service level improvement, cost and cash turnover reduction, etc. These advantages of cross-docking makes firms more adaptive to nowadays customers various choices and consumption habits, where smaller volumes of good, faster and more frequently deliveries are required [21,24].

Unlike the early implementation of cross-docking in the transportation industry, it has recently attracted more attention of researchers, being more than 85% of the published papers in cross-docking from 2004 until now. Two review papers have been recently published in cross-docking. J. Van and Valckeneaers et al. [16] provide an overview of the cross-docking concept and discuss guidelines to implement cross-docking

successfully. The authors also described several characteristics between different cross-dock types and indicate several opportunities to promote current research. Agustina and Lee et al. [26] provide a comprehensive review of three different types of models in cross docking, namely operational, tactical, and strategic models. Of particular interest to our work, Boysen and Fliedner [4] provide a detailed review to classify deterministic truck scheduling which helps to point out future research directions.

Given that cross-docking is a complex logistic strategy, there are several important requirements so as to be implemented successfully. Both software and hardware are indispensable. With the development of technology, the application of hardware like material handling devices and sorting systems is more and more common in logistics networks. However, in terms of software, computational professionals who are able to design the specific software system to apply different cross-docking configurations are in severe shortage, which hinders the implementation of cross-docking [13]. The information flow plays a significant role in cross-docking as compared to the traditional distribution center. For example, the arriving time of trucks and the amount and type of products they carry should be known before they arrive at cross-docks in order to be scheduled to appropriate docks. Information technologies (e.g. electronic data interchange, shipping container marking, bar-coding and scanning of products using universal product codes) provide accurate information to ensure the effectiveness of

cross-docking [3].

Considering the above conditions, cross-docking is only applicable to a certain kind of products with specific characteristics. According to Apte and Viswannathan [3], firstly, the products such as commonly used grocery products, regularly consumed perishable food, and chilled goods are more suitable for cross-docking. Those products have demand rates which are more stable and constant, thus the warehousing and transportation requirement of products are much more predictable and the planning and implementation of cross-docking become easier. Secondly, the products should have low unit stock-out cost. Cross-docking reduces the inventory cost. On the other hand, it has higher probabilities of stock-out due to the nature of cross-docking. However, if the unit stock-out cost is low, the benefits of reduced inventory cost can outweigh the stock-out cost. As shown in Figure 2, the products that have stable and constant demand rate and low unit stock-out cost are more suitable to use in cross-docking. The traditional

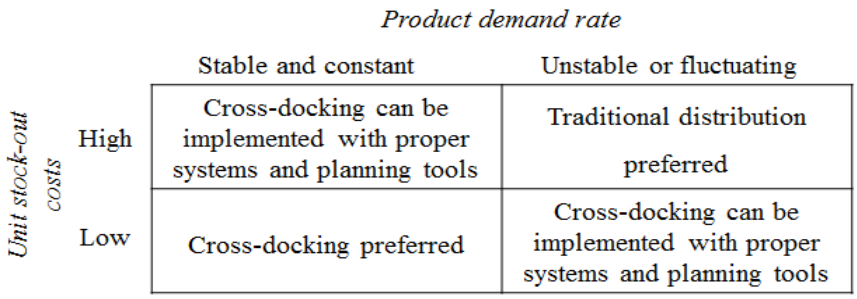

Figure 2. Suitability of cross-docking (adapted from Apte and Viswannathan [3])

warehousing is still preferable for the products with unstable demand and high unit

stock-out cost. However, cross-docking can still be used in the latter two situations if there are more precise planning systems.

2.2 Decisions in Cross-docking

The problems faced by cross-docking decision makers range from the longer term decisions (strategic or tactical) to the short term decisions (operational). This section reviews each class of problems separately. Strategically, the very first problems are the location and designing layout of cross-docks as well as the design of the entire cross-docking network. Once these decisions are taken, decision makers have to make sure that goods flow fluently through the network. Next comes to operational decisions like vehicle routing (how vehicles pick up the goods from the suppliers and send them to cross-docks at minimum cost) and truck scheduling (the assignment of trucks to dock doors). Other decisions such as how to manage internal resources for the loading and unloading of goods are also relevant.

2.2.1 Strategic Decisions: Location of Cross-docks

The location of one or more cross-docks plays an important role in the whole supply chain. Sung and Song [28], Gümüş and Bookbinder [29] are among the first researchers to study the location of cross-docks. Sung and Song [28] present a path-based integer programming formulation for the considered problem, which is similar to the formulation

provided by Donaldson et al. [30] and Musa et al. [31], based on similar simplifying assumptions. To solve the problem, the authors proposed a tabu search (TS) algorithm and use strong valid-inequalities to the proposed formulation. Computational experiments show that the TS algorithm works very well and valid-inequalities provide strengthened lower bounds. The previous work is extended by Sung and Yang [32] with an improved TS algorithm. The authors also develop an exact branch-and-price algorithm based on a set-partitioning formulation. Gümüş and Bookbinder [29] consider direct shipments and multiple product types. They solve some small instances with a mixed integer program using general purpose solvers. The authors conclude from the analysis of the experiments that the optimal number of cross-docking is an increasing function of the ration between the truck cost and the facility set up cost.

2.2.2 Strategic Decisions: Layout of Cross-docking Terminals

Another important strategic decision is the design (extern shape and intern layout) of the cross-dock terminal. As the labor cost is one of the most costly aspects in a cross-docking terminal, the design of a good layout so that trailers can be assigned to doors smoothly to improve the efficiency of the workers is very important. In [33], the authors design the layout of a cross-dock to minimize the labor cost of delivering goods by modeling travel costs and reducing three types of congestion typically experienced in a cross-dock. By doing so, they are able to improve the productivity of a cross-dock in Stockton, CA by

more than 11%. The same authors extend their work by studying the shape of a cross-dock [33]. They conclude that, for small to mid-size cross-docks, the most commonly shape (i.e. narrow rectangle or I-shape) works best and L or U shape should be avoided due to the possible cost of additional corners. However, for the cross-docks that have around 150 to 200 docks, the T-shape is the most suitable. An X-shape is preferred for larger cross-docks (more than 200 docks), for the reason that cross-docks with X-shape have greater centrality.

2.2.3 Tactical Decisions: Network Flow Optimization

As soon as the location and layout of cross-docks are decided, the cross-docking decision makers have to deal with the problem of efficiently transshipping goods among cross-docks and other supply chain participants (e.g. suppliers and customers).

Compared to the traditional transshipment problem, Lim et al. [35] take shipping and delivery time as well as transportation constraints into consideration and employ just-in-time objectives in a cross-docking network. The authors provide models to minimize the cost of inventory and schedules in transshipment networks with time-constrained supply and demand and transportation schedule constraints. Optimality conditions are provided for some polynomially solvable cases. For other cases, the authors prove that the problem is *NP*-hard. Chen et al. [36] study a similar problem which they call the multiple cross-dock problem, but unlike Lim et al. [35], they assume that

freights specified for a single delivery or pickup are not allowed to be split during the distribution process. The authors also consider a multi-commodity flow version of the problem, which is proved to be *NP*-complete. The authors provide an integer programming formulation and develop three heuristics, namely simulated annealing, TS and a hybrid of both to solve the considered problem. Among these heuristics, TS works better. Compared to the integer programming formulation solved by CPLEX, heuristics use less than 10% of the CPU time. Using a different approach, Donaldson et al. [30] consider the shipment of goods as individual transportation units instead of flows to benefit from consolidation. The authors develop an algorithm which is similar to a branch and bound algorithm after the latter fails to solve the problem in reasonable CPU times. Musa et al. [31] tackle the same problem with an ant colony optimization heuristic and report better results as compared with a branch and bound algorithm solved by LINDO in terms of CPU time and solution quality.

2.2.4 Operational Decisions: Vehicle Routing

We next discuss cross-docking from an operational point of view. Vehicle routing problems (VRP) arise when goods come from various locations and end in different destinations. Even though there is huge number of papers dealing with VRP [25], few of them deal with cross-docking and vehicle routing together. Probably, Lee et al. [38] is the first paper to tackle this problem. The authors propose a TS algorithm to find routing

schedule for pickup and delivery to minimize the sum of transportation cost and fixed cost of vehicles. To test the quality of the proposed algorithm, they compare with the solution obtained with an enumeration procedure and the results show the proposed algorithm works well with at most a 4% deviation of total cost. Liao et al. [39] work on the same problem with a new TS algorithm and they report the average improvements are as high as 10 to 36% for different size of problems as compared with TS algorithm proposed by Lee et al. [38]. Wen et al. [40] study the Vehicle Routing Problem with Cross-Docking. A fleet of homogeneous vehicles pick up products from suppliers. The products are consolidated at the cross-dock and then sent to destinations or customers immediately by the same set of vehicles, without storing at the cross-dock. The aim is to minimize the total traveled distance within a time window constraint. The authors present a mixed integer linear programming formulation which has a large number of variables and constrains and a TS algorithm. They test the proposed algorithms using real data from Transvision and computational results show that the algorithm produces good quality solutions for both small (less than 1% away from the optimum) and large instances (less than 5% gap with a lower bound) within very small CPU times.

2.2.5 Operational Decisions: Dock Door Assignment

As soon as trailers arrive at a cross-dock, they have to be assigned to doors as well. A good assignment can reduce dock delay and operational costs (e.g. pickup, delivery and

drivers cost). An early study was conducted by Peck [41] who develops a simulation model to model the assignment of trucks to dock doors. A greedy balance algorithm is also proposed to minimize the travel time of the shipments. Simulation results show the decisions made by the heuristic algorithm outperform those based on experience and intuition. Tsui and Chang [42] propose a bilinear program to determine the assignment of trucks to dock doors. Due to the fact that up-to-date data is usually difficult to obtain, their models are unable to provide the optimal solution for specific cases. But the decision makers can still use their solutions as a good starting point and modify models (e.g. add more constrains) for specific cases. Bermúdez and Cole [43] modify the model presented by Tusi and Chang [42] to serve cases that an origin or destination zone needs more than one door. They develop a genetic algorithm and compare it with pairwise exchange technique with the data from a less-than-truckload (LTL) logistic provider. Other works such as Bozer and Carlo [44], Bartholdi and Gue [45], Yu et al. [46] consider the dock door assignment from a semi-permanent layout point of view.

2.2.6 Operational Decisions: Truck Scheduling

Truck scheduling problems consider that there are not enough dock doors available for the arrival of incoming trucks. Thus, trucks have to wait in the cross-dock yard until planners decide when and which doors trucks should be assigned to. Because of the inherent complexity, researches started to study truck scheduling from simplified cases

(one strip dock and one stack dock). In a cross-dock scheduling review paper, Boysen and Fliedner [4] classify deterministic truck scheduling to structure and promote scientific progress on the field.

2.4.6.1 Single Strip and Stack Door

Yu [47] is probably one of the first works dealing with truck scheduling problems in cross-docking. In his Ph.D. dissertation, the author identifies thirty-two different models based on different cross-docking settings. Among those thirty-two models, three of them are focused to study the case where only one strip dock and one stack door are considered. In the first model, temporary storage is allowed and inbound trucks and outbound trucks cannot leave docks until all products are loaded and unloaded. In the second model, temporary storage is not permitted so that products have to move from the inbound dock to outbound dock immediately. However, inbound trucks and outbound trucks can leave and return to docks during the operations. In the last model, temporary storage is allowed and inbound trucks and outbound trucks are allowed to move out and in during the operations too. The goal of the problems is to minimize the makespan. To solve the considered problems, the author presents different approaches: mixed integer programming formulations, complete enumeration procedures, heuristic algorithms based on different dispatching rules, and a branch and bound algorithm. Although the first two are able to obtain optimal solutions, it is computationally expensive when instances

become larger. Therefore, for the large size instances, heuristics perform better in terms of time without sacrificing much on solution quality. However, the instances are generated randomly which may be not realistic. Moreover, the largest considered instance has only 6 inbound and outbound trucks with 9 types of products which is not really a large instance in today cross-docking systems.

With the same restrictions of the first model presented by Yu [47], Vahdani and Zandieh [48] propose five metaheuristic algorithms for the same problem: a genetic algorithm, a TS algorithm, a simulated annealing algorithm, an electromagnetism-like algorithm, and a variable neighborhood search algorithm. They use solutions from Yu as initializations for five metaheuristic and results on computational experiments report their improvement. For the same problem, Arabani et al. [49] also apply five metaheuristic algorithms: a genetic algorithm, a tabu search algorithm, a particle swarm optimization algorithm, an ant colony optimization algorithm, and a differential evolution algorithm.

Boysen et al. [11] address a truck scheduling problem which is very similar to the one studied in [47] with one inbound dock and one outbound dock and storages buffer to hold items temporarily. However, they handle the problem in a more aggregate view instead of a detailed scheduling. They propose an integer programming formulation and prove the problem is strongly *NP*-hard. A decomposition approach is developed, where the original problem is solved by decomposing it into two sub-problems. For each sub-problem,

either a fixed inbound sequence or a fixed outbound sequence is given and the optimal sequence of the other is obtained by an exact bounded dynamic programming approach. A priority rule-based heuristic to start the procedure is also presented. To tackle the original problem, the two sub-problems are solved iteratively until some stopping criteria is met. With the optimal solutions obtained by complete enumeration for small size instances, the performance of the decomposition approach for the overall problem can be evaluated and computational experiments show the proposed algorithms can provide high quality solutions with small CPU times. Nevertheless, they do not present any computational results for the proposed integer programming formulation with a general purpose solver (such as CPLEX). They use a set of small size instances with a particular structure to assess the efficiency of their algorithms.

Chen and Lee [19] model the truck scheduling problem as a two-machine flow shop problem, where two machines can be considered as the inbound dock and the outbound dock; unloading tasks for incoming goods can be viewed as jobs on the first machine; loading tasks for outgoing goods can be viewed as jobs on the second machine. They assume that certain products in some set of inbound trucks have already dedicated to a specific outbound truck so that an outbound truck cannot leave until all the corresponding inbound trucks have been unloaded. They prove the problem is strongly *NP*-hard and observe some properties that are helpful to solve the problem. Two polynomially solvable

special cases are presented. Finally, they propose a branch and bound algorithm to obtain optimal solutions with up to 60 trucks in reasonable CPU times. However, the authors do not consider the delivering time inside cross-docks from the inbound dock to the outbound dock.

Some other papers address similar problems. Forouharfoard and Zandieh [50] aim to minimize the number of products that pass through temporary storage in a cross-dock. Vehdani et al. [52] study a similar problem, however, they do not allow temporary storage. Soltani and Sadjadi [53] develop two metaheuristics, a genetic algorithm and an electromagnetism-like algorithm, to solve the same problem as Vehdani et al [52]. When considering the scheduling of outbound trucks, Lardi et al. [54] handle a single strip and a single stack door cross-dock scheduling problem under three scheduling scenarios considering that different amount of information is known in advance, e.g. the sequence and the content of all inbound trucks are known. To solve the first case, an optimal graph based model is presented and, for the other two cases, some heuristics are developed. Alpan [55] extends the problem to the case of multiple strip and stack doors and proposes a graph-based dynamic programming approach to solve the problem optimally.

2.4.6.2 Scheduling of Inbound Trucks

In order to study more realistic cross-docks, some papers deal with the scheduling of the inbound trucks assuming that the outbound trucks are already assigned to stack doors.

Rosales et al. [56] reduce the cost and provide a better workload balance to all workers in one shift for a large cross-docking in Georgetown by using a mixed integer programming formulation. Wang and Regan [57] provide two time-based algorithms (processing and transferring times) for the same problem and perform a simulation study to compare both algorithms. Computational experiments show the proposed time-based rules can save large amounts of time. Acar et al. [58] work on a variant of the problem that assumes truck arrival times are uncertain. They formulate the problem as a mixed integer quadratic program. Due to the complexity of the formulation, they develop a heuristic algorithm. McWilliams et al. [12] focus on the minimization of the makespan in a parcel hub. The authors develop a simulation-based scheduling algorithm to solve the problem with a significant reduction in the makespan by 4.2% to 35.8%.

2.4.6.3 Scheduling of Inbound and Outbound trucks

Lim et al. [60] consider the scheduling of both inbound and outbound trucks. They provide an integer programming formulation and propose two metaheuristics to solve the problem. Lim and Miao et al. [62] extend this work by taking transportation times into account to minimize operational cost and unfulfilled shipment. Compared with CPLEX, the proposed metaheuristics outperform in terms of solution quality and running time. Boysen [63] tackles truck scheduling for a cross-dock without allowing temporary storage. A dynamic programming and a simulated annealing procedure are presented.

Computational experiments report that high quality solutions can be obtained by the proposed approaches. A recently study conducted by Kuo [64] introduces a problem that deal with the assignment and sequencing of both inbound and outbound trucks in a multiple strip and stack docks environment.

Chapter 3: The Truck Scheduling Problem

In this chapter the formal definition of the *truck scheduling problem* (TRSP) in cross-docks, introduced by Boysen et al. [11], is first presented. The assumptions and mathematical notation required for formulating the TRSP are then summarized. In the Section 3.3, two mathematic programming formulations are shown, where the first one is provided by Boysen et al. [11] and the second one is an alternative formulation introduced in this thesis.

3.1 Problem Definition

We consider the schedule of a set of I inbound trucks to a single strip door and a set of O outbound trucks to a single stack door of a cross-dock terminal. To simplify the problem, all inbound trucks and outbound trucks are assumed to be available at the beginning of the planning horizon. Units of different products $p \in P$ are carried by each truck. Let a_{ip} denote the number of units of type p in an $i \in I$ inbound truck and let b_{op} denote the number of units of type $p \in P$ product required by the $o \in O$ outbound truck. We assume the requirement meets supply so the total number of products in all inbound trucks equals to the total number of products required by all outbound trucks. Thus, the following equation holds: $\sum_{i \epsilon I} a_{ip} = \sum_{o \epsilon O} b_{op} \ \forall p \in P$.

As trailers are usually homogeneous and cross-docking aims at moving only full loads

(e.g. mail distribution systems), the handling times for different trailers do not strongly differ in most cases. Therefore, it is realistic to assume that, in a same service slot (period) t, all products in an assigned inbound truck (or an assigned outbound truck) are unloaded (or loaded), where all handling operations (e.g. docking, unloading, undocking) required to process the truck are completed within this time span.

Once unloaded, the products have to be moved from strip to stack doors going through several stages inside the cross-dock. These stages include electronic scanning, quality inspection and coordination. The movement process is assumed to have a fixed movement time m. However, the actual movement can either start immediately for any unloading unit (e.g. using conveyer belt systems) or after completely unloading all products from the inbound truck (e.g. a worker stacks all units behind the receiving door before moving them). In the first case, the time span becomes $t + m$. In the second case, the time span becomes $t + m + 1$ as 1 represents the unit of time to wait for the inbound truck to be completely unloaded. Now, no matter m or m+1, the operational time can be ignored in the model of problem since after obtaining a solution, a proper re-indexing of slots outbound trucks are assigned to guarantee the exact final decision of the outbound trucks schedule. Once the movement process is completed, the products may wait in a temporary storage of enough size until they are loaded to outbound trucks.

The TRSP determines the sequences of inbound and outbound trucks to be assigned to a

single trip and a single stack door, respectively, in order to minimize the total completion time (i.e. makespan). If the sequence of the inbound trucks is fixed and the goal is to only schedule outbound trucks, the TRSP reduces to the sub-problem OUTBOUND – TRSP. Similarly, when the sequence of the outbound trucks is fixed and the goal is to only schedule inbound trucks, the TRSP reduces to the sub-problem INBOUND - TRSP.

3.2 Assumptions and Notation

The summary of the assumptions to model the problem is listed as follows:

1. There are only one receiving door and one shipping door in the cross-dock and there are located at different places of the terminal (segregated mode of service).
2. The time of processing (i.e. loading or unloading processes) for each truck is the same.
3. All inbound trucks and outbound trucks are available at the beginning of the time horizon. There are no predefined restrictions on truck assignments to slots (e.g. release or due dates)
4. The input data is known in advance and deterministic.
5. The time for delivering products from the receiving door to the shipping door is constant and therefore can be ignored.
6. The numbers of the product in the inbound trucks are equal to the numbers of products required by the outbound trucks.

7. The size of the temporary stock is unlimited.
8. Any combination of the sequences of the inbound trucks and outbound trucks represents a feasible solution.

The following notation is used in the mathematical programming formulations:

Input Data:

I: Set of inbound trucks (index i)

O: Set of outbound trucks (index o)

T: (Maximum) number of time slots available for (un-)loading trucks (index t)

P: Set of products (index p)

a_{ip}: Quantity of product type p arriving in inbound truck i

b_{op}: Quantity of product type p to be loaded onto outbound truck o

m: Movement time of products across the dock (w.l.o.g., $m = 0$)

3.3 Mathematical Programming Formulations

With the assumptions and notation at hand, the scheduling of the inbound and outbound trucks can be easily transformed to the sequence of inbound and outbound trucks. The objective is to obtain a sequence of trucks such that the makespan is minimized. The makespan consists of the time span from the time the first inbound truck is assigned to the time the last outbound truck is assigned. The following sets of decision variables are

defined:

$$x_{it} = \begin{cases} 1, \text{if inbound truck } i \text{ is assigned to slot } t \\ 0, \text{otherwise} \end{cases}$$

$$y_{ot} = \begin{cases} 1, \text{if outbound truck } o \text{ is assigned to slot } t \\ 0, \text{otherwise} \end{cases}$$

Using these variables, Boysen et al. [11] formulate the problem as follows (F1):

Minimize $C(X,Y) = C_{max}$ (1)

Subject to:

$C_{max} \geq y_{ot} \cdot t \quad \forall\, o \in O;\ t = 1, \dots, T$ (2)

$\sum_{t=1}^{T} x_{it} = 1 \quad \forall\, i \in I$ (3)

$\sum_{i \in I} x_{it} \leq 1 \quad \forall\, t = 1, \dots, T$ (4)

$\sum_{t=1}^{T} y_{ot} = 1 \quad \forall\, o \in O$ (5)

$\sum_{o \in O} y_{ot} \leq 1 \quad \forall\, t = 1, \dots, T$ (6)

$\sum_{\tau=1}^{t} \sum_{i \in I} x_{i\tau} \cdot a_{ip} \geq \sum_{\tau=1}^{t} \sum_{o \in O} y_{o\tau} \cdot b_{op} \quad \forall\, t = 1, \dots, T; p \in P$ (7)

$x_{it} \in \{0,1\} \quad \forall\, i \in I;\ t = 1, \dots, T$ (8)

$y_{ot} \in \{0,1\} \quad \forall\, o \in O;\ t = 1, \dots, T$ (9)

The objective function (1) and Eq. (2) compute the makespan, which is the time slot of the last assigned outbound truck. Eq. (3) ensure that every inbound truck is assigned to exactly one time slot and constraints (4) enforce that at most one truck can be assigned to a certain time slot. Analogously, Eq. (5) and (6) state the same idea for the outbound trucks. Constraints (7) ensure that an outbound truck can be assigned to a slot *t* only when all the required demand of that outbound truck can be satisfied by the remaining products

in the temporary stock, which is all the products sent by previous inbound trucks except the products that have already delivered by preceding outbound trucks.

As the goal is to minimize the makespan, the number of required service slots remains unknown until the solution of the model. Therefore, in order to solve the problem, we always initialize the number of slots T with some upper bound $\bar{C}$ on the makespan: $T = \bar{C}$. As $\bar{C}$ dramatically affects the number of variables and constraints, the determination of $\bar{C}$ is very important. In light of this, a simple upper bound can be constructed in the case of the worst scenario when the last scheduled inbound truck carries a product that is required by the first scheduled outbound truck, i.e. $\bar{C} = |I| + |O| - 1$. Furthermore, the following property of optimal inbound schedules can be used in order to tighten the formulation when solved with a general purpose solver.

Left-shift property [11]: Whenever an optimal solution exists, $|I|$ inbound trucks are always assigned to the first $|I|$ slots, even if the sequence is unknown. With this property, the number of variables and constraints can be reduced.

We next present an alternative formulation for the problem. We define additional integer decision variables g_{itlp}, which denotes the number of products of type p coming from truck i moved from time slot t (receiving door) and shipped by an outbound truck in time slot l ($t <= l$). The reason behind adding extra decision variables is mainly to improve its associated linear programming relaxation bound.

The problem can be thus formulated as follows (F2):

$$\text{Minimize } C(X,Y) = C_{max} \tag{10}$$

Subject to:

$$C_{max} \geq y_{ot} \cdot t \qquad \forall\ o \in O;\ t = 1, \dots, T \tag{11}$$

$$\sum_{t=1}^{T} x_{it} = 1 \qquad \forall\ i \in I \tag{12}$$

$$\sum_{i \in I} x_{it} \leq 1 \qquad \forall\ t = 1, \dots, T \tag{13}$$

$$\sum_{t=1}^{T} y_{ot} = 1 \qquad \forall\ o \in O \tag{14}$$

$$\sum_{o \in O} y_{ot} \leq 1 \qquad \forall\ t = 1, \dots, T \tag{15}$$

$$\sum_{l=t}^{T} g_{itlp} = a_{ip} x_{it} \qquad \forall\ i = I;\ p \in P;\ t \in T \tag{16}$$

$$\sum_{i \in I} \sum_{t=1}^{l} g_{itlp} \geq b_{op} y_{ol} \qquad \forall\ p \in P;\ o \in O;\ l \in T \tag{17}$$

$$x_{it} \in \{0,1\} \qquad \forall\ i \in I;\ t = 1, \dots, T \tag{18}$$

$$y_{ot} \in \{0,1\} \qquad \forall\ o \in O;\ t = 1, \dots, T \tag{19}$$

The objective function (10) and constraints (11-15) have the same meaning as in the previous formulation (F1). Eq. (16) guarantee that all products from any inbound trucks are delivered to the shipping door. Constraints (17) ensure that at time slot *l*, there are enough products for the assigned outbound truck.

Chapter 4: Solution Methods

In this chapter we present solution algorithms for the TRSP. The necessity to develop specialized methods arises not only from the fact that the TRSP is *strongly NP*-hard [11], but also because general purpose solvers can only solve (relatively easy) small-size instances. We first present two lower bounds strategies introduced by Boysen et al. [11]. These can be used to provide an estimation of quality of the solutions obtained with the proposed heuristic algorithms. We then present four heuristic algorithms to obtain feasible solutions for the TRSP: a *local search (LS),* a *simulated annealing (SA),* a *large neighborhood search (LNS),* and a *beam search (BS).*

4.1 Lower Bounds

Due to the fact that all inbound and outbound trucks have to be scheduled at some time slot, the first trivial lower bound $\underline{C}^1$ for the optimal solution value of the TRSP can be obtained as follows:

$$\underline{C}^1 = \max \{|I|; |O|\}.$$

To construct another lower bound $\underline{C}^2$, the overall problem is divided into $|P|$ sub-problems. For each product, $p \in P$, inbound and outbound truck sequences are constructed by considering the following steps:

- The set of $|I|$ inbound trucks are sorted in descending order with respect to loads a_{ip} of the considered product p. A sequence vector π^p with elements π_i^p (i = 1,…, $|I|$) denotes the sorted inbound trucks sequences. Because of the left-shift property, the first truck is to be scheduled at the time slot $t = 1$.
- The set of O outbound trucks are sorted in ascending order with respect to loads b_{op} of the considered product p. A sequence vector μ^p with elements μ_o^p (i = 1,…, $|O|$) stores the sorted outbound trucks sequences. The total time slots s_{op} for each product p can be computed according to the following equations:

$$s_{op} = \min\left\{t = s_{o-1\,p} + 1, \dots, T \,\middle|\, \sum_{\tau=1}^{\min\{|I|;t\}} a_{\pi_\tau^p p} \geq \sum_{\tau=1}^{o} b_{\mu_\tau^p p}\right\} \quad \forall\, o \in O;\ p = P$$

To initialize the recursive formulae, a slot μ_o^p has to be initialized with slot number 0.

The maximum makespan associated with all products leads to the final lower bound $\underline{C}^2$:

$$\underline{C}^2 = max_{p \in P}\{s_{|O|p}\}$$

4.2 Local Search

LS is first reported to be successfully implemented in combinatorial optimization problems by Croes in 1958 [61] and has ever since become one of the most frequently and widely used heuristics in the last 50 years. It starts with a feasible solution and aims to improve it by generating a new solution that is close to the current solution. For that, a neighborhood is a set of solutions that are close to a given solution. Then the best

solution in the neighborhood is identified and it replaces the current solution and the procedure is repeated. However, if the solution does not improve, the iteration stops and the current best solution is said to be local optimal.

To describe the features of a LS algorithm, we define the following notation. Let *S* be a solution to the problem, which represents a sequence of inbound trucks and outbound trucks. Let *N(S)* denote the set of solutions with elements S_i, which are the neighbors of the solution *S*, and let S_i' be the neighbor that has the minimum makespan. Based on the current sequence *S*, the elements in *N(S)* are constructed by fixing the position of some trucks and changing the position of other trucks. *F(S)* is the cost function which is the makespan of a specific sequence *S*. A general LS algorithm is depicted in Algorithm 1.

Algorithm 1 Local Search

Let *F(S)* be the function to minimize, *S* some initial feasible solution and *N(S)* the neighborhood structure.

StopCriterion ← **false**

While (StopCriterion = false) **do**

Search for a solution $S_i' \in N(S)$ with $F(S_i') < F(S)$

If ($F(S_i') >= F(S)$,) **then**

StopCriterion ← **true**

Else

$S \leftarrow S_i'$

End if

End while

A current solution is represented by $S = (a, b)$, where a: $T \rightarrow I$, is the inbound sequence mapping, i.e., $a(t) = i$ if inbound truck $i \in I$ is placed in time slot $t \in T$, and b: $T \rightarrow O$, is the outbound sequence mapping, i.e., $b(t) = o$ if inbound truck $o \in O$ is placed in time slot $t \in T$. In our implementation of the LS, we define two neighborhoods. The first one is the inbound truck neighborhood:

$$N_I(S) = \{S' = (a', b): \exists! (i_1, i_2), a(i_1) = a'(i_2), a(i_2) = a'(i_1), i_1 \neq i_2\},$$

which is obtained by swapping two adjacent or nonadjacent inbound trucks in a with outbound trucks in b fixed. The second one is the outbound truck neighborhood:

$$N_O(S) = \{S' = (a, b'): \exists! (o_1, o_2), b(o_1) = b'(o_2), b(o_2) = b'(o_1), o_1 \neq o_2\},$$

which is obtained by swapping two adjacent or nonadjacent outbound trucks in b with inbound trucks in a fixed.

We implement our LS as follows. We start the algorithm with a sequence of inbound and outbound trucks generated randomly. Preliminary computational experiments revealed that the initial sequence has little influence for the solution performance. We first explore $N_I(S)$ using a best improvement strategy until no improved solution is found. An example of the exploration of $N_I(S)$ is showed in Figure 3.

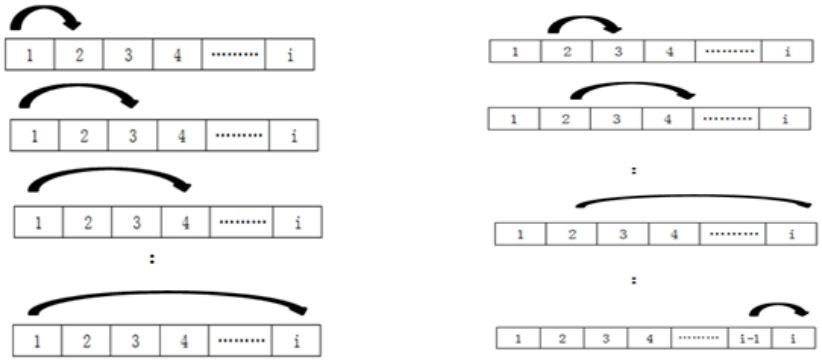

Figure 3. The first iteration and the second iteration for the initiate sequence

We then explore $N_o(S)$ using a best improvement strategy until no improved solution is found. We keep exploring $N_I(S)$ and $N_o(S)$ iteratively until no improved solution is found and the LS algorithm is terminated. An outline of the proposed LS algorithm is depicted in Algorithm 2.

Algorithm 2 Local Search

StopCriterion ← **false**

While (StopCriterion = false) **do**

Explore N_I

If (Solution not improved in N_I) **then**

Explore N_O

If (Solution has not been updated) **then**

StopCriterion ← **true**

End if

End if

End while

4.3 Simulated Annealing

SA, introduced by Černý [27] and Kirkpatrick et al. [34], is based on an analogy to the process of physical annealing with solids (Metropolis Algorithm), in which a crystalline solid is heated to a sufficiently high value and then cooled very slowly until all particles reach their most regular possible crystal lattice state. If the cooling schedule is sufficiently slow, in the finally state the solid is in a superior structural integrity and the energy of the system is minimal.

At each iteration of the SA, the cost function generates values for two solutions (the current solution and a newly selected solution). The incumbent is always replaced by the new selected solution if it has a better solution value, while a fraction of worse solutions are accepted in the hope of escaping local optima in search of global optima. The probability of accepting worse solutions depends on a temperature parameter, which is typically proportional to the difference in solutions and non-increasing with each iteration of the algorithm [37].

Simulated annealing starts with an initial solution *S*. Let S', which is randomly generated, be a neighbor of solution *S*. The candidate solution, S', is accepted as the current solution based on the following acceptance probability:

$$P\{\text{Accept } S' \text{ as next solution }\} = \begin{cases} exp\left[-\frac{F(S')-F(S)}{T_k}\right], & if\ F(S') - F(S) > 0 \\ 1, & if\ F(S') - F(S) \leq 0 \end{cases}$$

Define T_k as the temperature parameter at iteration k, such that

$$T_k > 0 \qquad \text{for all } k \text{ and } lim_{k\to\infty} T_k = 0$$

We implement our SA as follows. It starts with a feasible solution *S*. Preliminary computational experiments showed the using a good solution (e.g. a solution obtained with LS) has almost no impact on the final solution as compared to starting with a solution randomly generated. For that reason, we decide to use initial solutions randomly generated. We explore $S' \in N_I(S)$, where two inbound trucks are randomly selected and swapped their positions. The makespans of *S* and *S'* are computed and we replace *S* with *S'* if $w \geq exp\left[-\frac{F(S')-F(S)}{T_k}\right]$, where *w* is a value randomly generated between [0, 1] with uniform distribution and T_k is temperature parameter at iteration *k*; otherwise, we do not replace *S*. The above procedure is repeated *L* times. We then cool down that current temperature *k* times according to $T_k = rT_o$, where T_o is initial temperature and $r \in (0, 1)$. For each temperature T_k, we apply the same procedures described above to explore $N_I(S)$ and the same criteria to accept *S'*. The best solution in the end of iterations is the best sequence for a specific sequence of outbound trucks and with that best solution, we then explore $N_o(S)$ using the same procedure. We keep exploring $N_I(S)$ and $N_o(S)$ iteratively until no improved solution is found and the SA is terminated. An outline of the proposed

SA algorithm is depicted in Algorithm 3.

Algorithm 3 Simulated Annealing

Choose an initial solution S
While (StopCriterion = false) **do**
  Set an initial temperature T_o, a reduction factor $0< r <1$, $\boldsymbol{k} \leftarrow 0$, $T_k \leftarrow T_o$
  While (not yet frozen) **do**
    Count $\leftarrow$ 0
      While (count < L) **do**
        Pick a random neighbor $S' \in N_I(S)$
        $\Delta = F(S') - F(S)$
        If ($\Delta \leq 0$) **then**
          $S \leftarrow S'$
        Else
          Set $S \leftarrow S'$ with probability $e^{-\Delta/T_k}$
        End if
        Count $\leftarrow$ Count + 1
      End while
    $k \leftarrow k$-1, $T_k \leftarrow rT_k$ (reduce the temperature)
  End while
  Set an initial temperature T_o, a reduction factor $0< r <1$, $\boldsymbol{k} \leftarrow 0$, $T_k \leftarrow T_o$
  While (not yet frozen) **do**
    Count $\leftarrow$ 0
      While (count < L) **do**
        Pick a random neighbor $S' \in N_O(S)$
        $\Delta = F(S') - F(S)$
        If ($\Delta \leq 0$) **then**
          $S \leftarrow S'$
        Else
          Set $S \leftarrow S'$ with probability $e^{-\Delta/T_k}$
        End if
        Count $\leftarrow$ Count + 1
      End while
    $k \leftarrow k$-1, $T_k \leftarrow rT_k$ (reduce the temperature)
  End while
End while

4.4 Large Neighborhood Search

The LNS metaheuristic was first introduced by Shaw [61]. In LNS the neighborhood of a current solution is constructed by *destroy* and *repair* mechanisms, where part of the current solution is destroyed by a destroy strategy and then reconstructed by a repair strategy. To illustrate the destroy and repair strategies, consider the INBOUND – TRSP. A simple destroy strategy is to randomly remove a proportion of trucks scheduled in a given position of the sequence of the current solution and a simple repair strategy is to reconstruct the solution by randomly reassigning the removed trucks into the available positions of the sequence. The Figure 4 illustrates the destroy and repair mechanisms. The top figure shows an INBOUND – TRSP solution before the destroy step. The middle figure shows the solution after a destroy operation that removed eight trucks. The bottom figure shows the solution after the repair step.

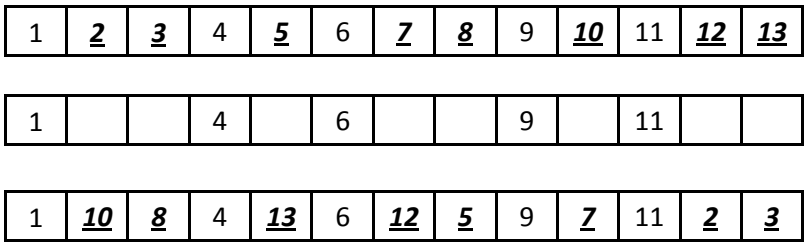

Figure 4. Destroy and repair example

The neighborhood of a destroy solution contains a large amount of solutions which explains the name of the algorithm. For example, we consider an INBOUND – TRSP with 32 trucks. If 60% of the trucks are to be removed, there are $C(32,20) = 32!/(20! \times$

12!) = 2.25 × 10^8 possible ways of doing it. To repair the solution, there are $A(32,20)$ = 32!/(20!) = 5.5 × 10^{26} different ways to do it [37].

To describe the details of the LNS algorithm, several definitions are introduced. Let S denotes the current solution, S^b the best solution obtained during the search, and S^t is a temporary solution. $F(S)$ is the cost function (makespan) of the solution S. The function $D(\cdot)$ is the destroy strategy, where $D(S)$ returns a partial solution of S. The repair strategy is represented by the function $R(\cdot)$ and $R(D(S))$ returns a repaired solution that was partly destroyed. The LNS starts with a feasible solution. Then, a new solution is obtained through the destroy and repair strategies. The temporary solution would be accepted in different criterions. A simply one is only to accept improving solutions. The best solution S^b would be updated if the cost function of S^t is smaller.

To design a more flexible algorithm, we borrow the acceptance criteria used in the SA introduced in the last section, where a temporary solution is always accepted if $F(S^t) <= F(S^b)$, and accepted with probability $exp\left[-\frac{F(S^t)-F(S^b)}{T_k}\right]$, if the cost function does not improve. The procedure to apply LNS is similar to the one of SA. Preliminary tests showed that using a good solution (e.g. a solution obtained with SA) has almost no impact on the final solution as compared to starting with a solution randomly generated. For that reason, we decided to use initial solutions generated randomly. The main difference of LNS is the strategy to choose the neighbor. The number of trucks removed

is called the degree of destruction. We apply different degrees of destruction, e.g. selecting and destroying five trucks and then replacing removed positions with trucks randomly selected from the removed trucks. An outline of the proposed LNS algorithm is depicted in Algorithm 4.

Algorithm 4 Large Neighborhood Search

Input: a feasible solution S

$S^b \leftarrow S$

Repeat

$S^t \leftarrow R(D(S))$

If accept (S^t, S) **then**

$S \leftarrow S^t$

End if

If $F(S^t) <= F(S^b)$ **then**

$S^b \leftarrow S^t$

Until stop criterion is met

Return S^b

4.5 Beam Search

BS was first developed by Lowerre in 1976 [59] for a speech recognition problem, where the goal was to obtain a solution quickly by searching a number of promising decision paths in parallel. BS is an adaptation of the well-known branch and bound (B&B) algorithm commonly used to solve integer programs. However, the requirements of CPU time and memory associated with B&B increase exponentially as the size of the instances

increase. BS, on the other hand, has a running time bounded by a polynomial that depends on the size of the problems and its parameters. The key idea of BS is to keep only some promising nodes and to permanently prune other nodes.

BS moves downward on the enumeration tree level by level from the best β promising nodes without backtracking. The other nodes are permanently discarded. To determine the best β promising nodes, there are typically two ways of doing it. One way is to apply an evaluation function which produces an estimation of the cost of a solution obtained from that partial solution. This evaluation function is called *one-step priority evaluation function*, which only considers the next decision to be made (the next job to schedule) and, thus, has a more local view. Another way is a total evaluation that uses some rules to construct a complete solution based on the current partial solution to estimate its cost. This evaluation has a global view of the solution. One can use either one or both strategies to apply BS. A *filtering* mechanism can combine both strategies together. During filtering, some nodes are discarded based on their local evaluation function values and only the remaining nodes are globally evaluated. The number of these remaining nodes is called the filter width (α).

We illustrate the main idea of BS through a truck scheduling example. There are five trucks needed to be sequenced. We schedule one more truck at each level. As shown in Figure 5, nodes represent partial schedules. There are no trucks scheduled at level 0 and

one truck has selected at level 1, and so forth. The total number of nodes that can be explored at each level is $\{(|I|!)/(|I| - |K|)!\}$, where I is the total number of trucks to be sequenced and K is the level number. A line linking two nodes represents the decision to add one more truck based on the partial schedule. The circles with dotted line represent the nodes selected by the local evaluation (the filter width). The solid circles are selected by the global evaluation to be further explored. The beamwidth in the Figure 5 is two so that, in the end of the enumeration tree, two feasible solutions are selected with associated sequences $\{i_1, i_2, i_3, i_5, i_4\}$ and $\{i_3, i_2, i_4, i_5, i_1\}$.

In our implementation of BS, we apply only a total cost evaluation by constructing complete solutions. Two main phases are taken. In the first phase, n best inbound

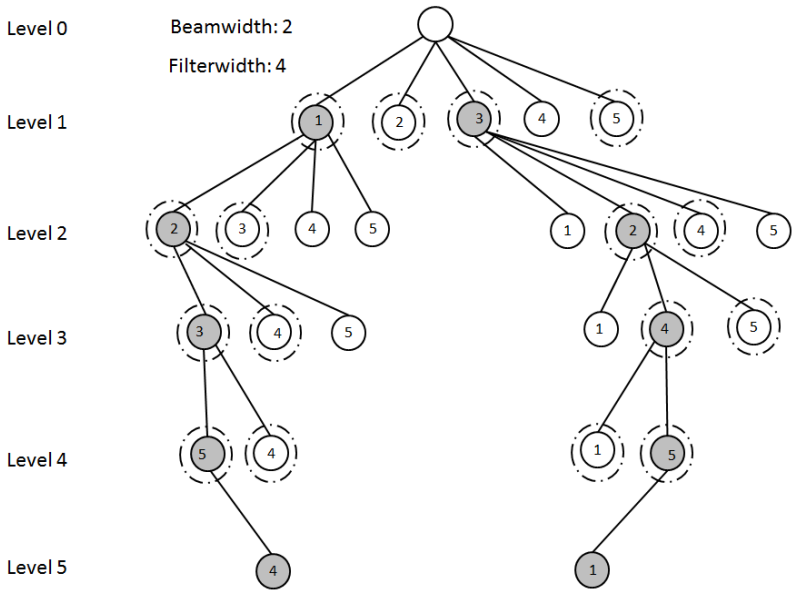

Figure 5. The beam search tree

sequences are selected. The algorithm starts by fixing the positions of inbound trucks depending on the level, e.g. fixing the first three trucks at level three, and builds the complete sequence with the same procedure used in the computation of the second lower bound strategy, i.e. sorting the rest of inbound trucks in descending order with respect to the load a_{ip} of the considered product p. Outbound trucks are sorted in ascending order with respect to loads b_{op} of the considered product p. For each node, there are $|P|$ different sequences with $|P|$ lower bounds associated and we pick the sequence that has the minimum makespan, or has the maximum lower bound value, or has the minimum difference between the makespan and lower bound, depending on which filtering approach is used later, to represent the complete sequence of that node. To calculate the global cost function, there are three different filtering approaches. The first approach is based on the makespan, the second is based on the lower bound, and the last one is based on the difference between the makespan and the lower bound. The three approaches follow the same processes, where we select n nodes with the least makespan (or lower bound or difference) at each level. If some nodes have the same makespan (or lower bound or difference), we select randomly some of them. The maximum number of nodes we can select at the first level equals to the number of inbound trucks $|I|$. However, when we move downwards to second level, the maximum number of nodes we can select is $|I| \times (|I|-1)$. Thus, we can define in theory the inbound beamwith to be a number between 1 and $|I| \times (|I|-1)$. In the second phase, we select the best m outbound sequences for each

sequence among those *n* best inbound sequences. For the outbound trucks at each level, we start fixing the positions of outbound trucks depending on the level, e.g. fixing the first 3 trucks at the level 3. We then sequence other outbound trucks in ascending order with respect to their fraction of total product volumes:

$$f(o) = \frac{1}{\sum_{p \in P} \frac{b_{op}}{\sum_{\sigma \in O} b_{\sigma p}}} \text{ (the fraction of total product volume).}$$

For example, there are three outbound trucks carrying three types of products (*a, b, c*). The first truck carries 2 units of *a*, 3 units of b and 4 units of *c*; the second truck carries 2 units of *a*, 3 units of b and 5 units of *c*; the third truck carries 6 units of *a*, 3 units of b and 5 units of *c*. The fraction of total product volume of the first truck is

$$f(1) = \frac{1}{\frac{2}{2+2+6} + \frac{3}{3+3+3} + \frac{4}{4+5+5}}$$

There is one complete sequence associated with each node. To calculate the global cost function, we use the makespan of each node, where we select *m* nodes with the smallest makespan at each level. If some nodes have the same makespan, we select among them randomly. We can define the outbound beamwidth to be a number between 1 and $|O| \times (|O|-1)$. At the last level of the enumeration tree, we have $n \times m$ feasible solutions. In addition, we also generate several feasible solutions during the evaluation of the nodes in the tree. We compare all these solutions and the sequence with the minimum makespan gives us the best solution.

Chapter 5: Computational Experiments

In the following, a computational study is presented to evaluate the performance of the formulations and solution methodologies introduced in previous sections. In the first part of this chapter, besides a set of instances randomly generated using the procedure presented in Boysen et al. [11], two more realistic sets of instances are generated. In the second part, the results obtained with the integer programming formulations presented in Chapter 3 are presented. We then analyze individually the performance of SA, LNS and BS introduced in Chapter 4. Finally, the results obtained with all proposed algorithms and CPLEX are compared. All algorithms were coded in C and run on Windows with a Pentium Dual-Core processor at 2.80 GHz and 4GB of RAM.

5.1 Instance Generation

Preliminary computational experiments showed that the instances generated in Boysen et al. [11] tend to be rather easy. In particular, all generated instances, containing up to 18 inbound and outbound trucks, can be solved by CPLEX within a few minutes (i.e. always less than 13 minutes). For that season, following the structure of real applications, we construct two new sets of instances to better assess the complexity of the TRSP and the performance of the proposed heuristic algorithms, for obtaining high quality feasible solutions in reasonable CPU times.

5.1.1 First Set of Instances

The first way to generate instances is based on the situation that each inbound truck carries only one type of product and outbound trucks carry a mixture of products. Table 1 shows an instance with 5 inbound and outbound trucks with 5 types of product (*a, b, c, d, e*).

Type of Product / Truck	a	b	c	d	e
Inbound trucks 1	**50**	0	0	0	0
2	0	**30**	0	0	0
3	0	0	**40**	0	0
4	0	0	0	**20**	0
5	0	0	0	0	**60**
Outbound trucks 1	17	13	0	9	18
2	0	0	18	0	14
3	18	0	13	7	13
4	0	17	0	2	15
5	15	0	9	2	0
Total units	50	30	40	20	60

Table 1. An example of a randomly generated instance

Two sets (i.e. small and large sized) of instances are generated using this approach. The parameters of numbers of inbound trucks and outbound trucks with their loads are shown in Table 2**.** There are 5 different numbers of inbound and outbound trucks for each set of instances so that $5 \times 5 = 25$ instances are generated. Each instance is generated as follows according to the given set of parameters.

		Values	
Symbol	Description	Small	Large
$\|I\|$	Number of inbound trucks	14, 16, 18, 20, 22	24, 26, 28, 30, 32
$\|O\|$	Numbers of outbound truck	14, 16, 18, 20, 22	24, 26, 28, 30, 32
$\|P\|$	Number of products	14, 16, 18, 20, 22	24, 26, 28, 30, 32
TF	total amount of product units in all inbound and outbound trucks for a family of products.	1000 - 9000	

Table 2. Parameters for instance generation

- Inbound trucks: Assuming that there are several origins (e.g. different product suppliers). Each origin provides one product and has only one truck carrying that product. The number of units of each product ranges from 1000 to 9000. The number of trucks is the same as the number of different types of products we have.
- Outbound trucks: For each product $p \in P$, the following procedure is repeated to generate the load of outbound trucks. We assume that, for each type of product, at least half of the outbound trucks carry one unit of it and it is decided by defining an array UO_p. The size of UO_p is decided by an equally distributed integer random number out of the interval [$|O|/2$, $|O|$]. The value of the element of UO_p is a randomly unrepeated integer that is chosen according to uniform distribution with the interval [1, $|O|$], which means we choose several outbound trucks to place in the array UO_p. Let TF_p be the total amount of product units of product p. We assume that each chosen outbound truck containing at least $\left\lfloor \frac{TF_p}{2\times|O|} \right\rfloor$ units of type p product. Let $redr_p^o$ be an equally distributed integer random number out of the interval [1, $\left\lfloor \frac{TF_p}{2\times|O|} \right\rfloor$]. The

chosen outbound truck may contain more products by adding $redr_p^o$. To avoid rounding errors we distinguish between the set *UOp*, which contains all randomly chosen trucks, and UO_p^-, which a copy of *UOp* missing its last element:

$$b_{op} = \begin{cases} \left\lfloor \frac{TF_p}{2 \times |O|} \right\rfloor + redr_p^o, & \forall\, p \in UO_p^- \\ TF_p - \sum_{p' \in UO_p^-} b_{op}, & p \in UO_p \setminus UO_p^- \end{cases}$$

5.1.2 Second Set of Instances

The second way to generate instances is based on the situation that several groups of inbound trucks carry several families of products from different origins and outbound trucks carry a mix of products. Table 3 shows an instance with seven inbound and five outbound trucks with three families of products (*a*, *b*, *c*).

The parameters of numbers of inbound trucks and outbound trucks are the same as the first set, though the load is now ranging from 1000 to 5000 for each inbound truck. We also generate two classes (small and larger) of instances for the second set with 25 instances for each class. Each single instance is generated according to the procedure as follows.

- Inbound trucks: There are several origins (e.g. different product suppliers). Each origin provides three to five products and has several trucks carrying units of each product. The number of units of each product randomly ranges from 1000 to 5000.

- Outbound trucks: the way to generate the load of outbound trucks is same as the first set of instances.

Type of product / Truck	a1	a2	b1	b2	b3	c1	c2
Inbound trucks 1	**50**	**10**	0	0	0	0	0
2	**20**	**30**	0	0	0	0	0
3	0	0	**10**	**30**	**10**	0	0
4	0	0	**20**	**30**	**20**	0	0
5	0	0	**40**	**10**	**30**	0	0
6	0	0	0	0	0	**50**	**10**
7	0	0	0	0	0	**60**	**40**
Outbound trucks 1	30	0	24	13	18	13	15
2	15	13	0	16	0	26	8
3	0	11	21	17	13	17	0
4	25	0	25	0	15	22	16
5	0	14	0	24	14	32	11
Total units	70	40	70	70	60	110	50

Table 3. An example of a randomly generated instance

5.2 Formulations and Three Sets of Instances

Preliminary computational experiments showed that the proposed formulation (F2) is able to improve the linear programming (LP) relaxation bound, but the required CPU time to solve the problem is much longer than the one required by formulation (F1). It seems that the improvement on the lower bound does not compensate the increase of the number of variables and constrains. However, we believe it is still useful to provide our

formulation (F2) due to the fact that with some decomposition technics (e.g. Lagrangean relaxation, column generation or bender decomposition) it would be possible to handle it and to solve the TRSP in reasonable CPU times. However, developing decomposition technics is not the scope of this thesis, we decide not to include the associated computational experiments for formulation (F2) and to only use the formulation (F1) to assess complexity of different structures of instances.

Three sets of instances are generated with the number of inbound trucks and outbound trucks ranging from 14 to 18, which are the largest size instances used by Boysen et al. [11]. The way to generate the first two sets is mentioned in Section 5.1 and the third set of instances is generated by following the procedure presented in Boysen et al. [11]. The detailed results of the comparison are provided in Table 4. The first and second columns contain the number of inbound and outbound trucks. The third column contains the number of different types of product. The last three columns correspond to the required CPU time in seconds to obtain an optimal solution for the three sets of instances.

The second set of instances require the least time to be solved, most of them in 60 seconds. The third set of instances require more time while the first set requires the most time. For the last instance of the first set, we cannot even obtain the optimal solution within 27 hours. Therefore, we conclude that first set of instances is the most difficult and the second set of instances is the easiest and the third set of instance goes in the middle.

It seems the first set is the most sensitive to the truck sequences, where only one inbound truck carries one type of products and changing the sequence of one truck affects the sequence of all inbound and outbound trucks. The second and third sets of instances are relatively easy because there are usually some trucks carrying the same types of products, where changing the sequence of one truck may not substantially affect the sequence of many inbound and outbound trucks.

Given that the third set of instances used in Boysen et al. [11] can be solved by CPLEX in few minutes, we decided to increase the size of instances (i.e. 26 to 30 inbound and outbound trucks) and to solve them with CPLEX to assess the complexity of the TRSP.

Truck Information			1st set		2nd set		3rd set	
InTruck	OutTruck	Product	OTP	Time(s)	OTP	Time(s)	OTP	Time(s)
14	14	14	23	90	16	28	15	26
14	16	14	25	60	19	11	17	96
14	18	14	26	195	22	14	18	34
16	14	16	25	7200	18	20	18	192
16	16	16	27	1500	20	25	18	776
16	18	16	29	1149	21	22	19	85
18	14	18	27	27420	19	61	20	95
18	16	18	29	510	20	30	19	402
18	18	18	31-30*	97230	23	34	19	482

Table 4. Comparison of results of three sets of instance running in CPLEX

The first three columns of Table 5 are the number of inbound trucks, outbound trucks and products. The following columns give the information of upper bounds and lower bounds

with associated gaps. These instances were run in CPLEX for 24 hours and the remaining gaps were still very large after that time. The size of trucks arranging from 26 to 30 are realistic in real life applications, but CPLEX cannot longer solve them to optimality. Therefore, it is very important to develop algorithms to obtain good quality solution in reasonable CPU times. We also test the larger instances generated by the second way and most of the instances can solved by CPLEX within few hours (i.e. three hours). However, our goal is to solve difficult instances so that we only run our proposed algorithms for the first and third sets of large instances. To compare the solution quality of all algorithms, we obtain the best known solutions from either the results of CPLEX for a given time frame of 3600 seconds or the best solution found with all proposed heuristic algorithms.

Truck Information			1^{st} set				3^{rd} set			
InTruck	OutTruck	Product	UB	LB	Gap(%)	Time(h)	UB	LB	Gap(%)	Time(h)
26	26	26	46	30	34%	24	36	22	39%	24
26	28	26	47	25	47%	24	40	20	50%	24
26	30	26	50	32	36%	24	45	22	51%	24
28	26	28	48	28	41%	24	40	26	35%	24
28	28	28	49	27	44%	24	49	27	44%	24
28	30	28	52	32	46%	24	39	19	51%	24
30	26	30	49	30	38%	24	49	20	59%	24
30	28	30	52	30	42%	24	49	25	48%	24
30	30	30	52	32	42%	24	35	14	14%	24

Table 5. Computational results of large instances solved with CPLEX

5.3 Evaluation of SA

In this section we present the computational results from the implementation of simulated annealing. Given that the cooling strategy has a big influence on the performances of SA, we test seven different cooling strategies. Table 6 shows the cooling parameters for different strategies, where InT_o and $OutT_o$ denote the initial temperatures for applying SA to INBOUND – TRSP and OUTBOUND – TRSP respectively, InT_f and $OutT_f$ are the final temperatures, the total numbers of temperatures which gradually cool down from the initial temperature to the final temperate are represented by *InK* and *OutK, InL* and *OutL* are the total number of iterations for each temperature. From the strategy 1 to the strategy 7, the speed to cool down the temperature becomes slower and the iterations in each cooling stage are 37, 111 or 150.

	S1	S2	S3	S4	S5	S6	S7
InT_o	1000	1000	1000	1000	1000	1000	1000
InT_f	1	1	1	1	1	1	1
InK	37	111	37	111	37	111	37
InL	5	2	10	5	10	10	150
$OutT_o$	1000	1000	1000	1000	1000	1000	1000
$OutT_f$	1	1	1	1	1	1	1
OutK	37	111	37	111	111	111	37
OutL	5	2	10	5	20	10	150

Table 6. SA cooling parameters

Table 7 shows the results of SA, where the percent deviation (PD) is measured by $\frac{C(SA)-C(OPT)}{C(SA)} \times 100\ \%$, where *C(SA*) is the best result from SA and *C(OPT)* is the best known solution. We only present the detailed results from a subset of instances, but the average PD and CPU time are computed by considering all instances.

Truck Information			SA1		SA2		SA3		SA4		SA5		SA6		SA7	
In Truck	Out Truck	Product	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)
1st set																
30	32	30	3.45	0.59	3.45	0.81	3.45	2.26	3.45	4.95	3.45	13.4	1.75	19.5	1.75	523
32	24	32	5.66	0.54	3.85	0.75	5.66	2.07	1.96	4.65	3.85	12.3	3.85	17.8	1.96	485
32	26	32	3.70	0.58	5.45	0.77	3.7	2.23	3.7	4.93	3.7	13.3	3.7	19.4	1.89	519
32	28	32	3.57	0.57	3.57	0.81	1.82	2.31	3.57	4.96	1.82	13.2	1.82	19.7	0	523
32	30	32	5.08	0.6	3.45	0.85	3.45	2.37	3.45	5.19	3.45	14.1	1.75	20.7	0	549
32	32	32	34.43	0.62	33.3	0.87	33.3	2.44	33.3	5.36	33.3	14.7	33.3	21.4	32.2	567
3rd set																
30	32	30	12.5	0.6	18.6	0.83	7.89	2.42	12.5	5.07	18.6	13.7	5.41	20.4	5.41	528
32	24	32	8.11	0.59	10.5	0.81	8.11	2.36	2.86	5.01	8.11	13.9	2.86	20.1	0	517
32	26	32	10.3	0.6	10.3	0.84	10.3	2.43	5.41	5.17	5.41	13.2	5.41	21.1	2.78	535
32	28	32	12.5	0.62	10.3	0.86	7.89	2.46	7.89	5.3	5.41	14.4	7.89	21.2	5.41	541
32	30	32	7.5	0.64	11.9	0.87	11.9	2.47	11.9	5.38	7.5	15.1	5.13	21.5	2.63	570
32	32	32	7.32	0.66	13.6	0.97	7.32	2.66	7.32	5.87	5	16	7.32	23.2	5	613
Average			**7.57**	**0.5**	**8.16**	**0.7**	**6.62**	**1.99**	**6.14**	**4.31**	**5.66**	**11.7**	**4.6**	**17.2**	**2.77**	**447**

Table 7. Results of SA with 7 different cooling strategies

As shown in Figure 6, from the strategy 1 to strategy 7, the trend of average CUP times increases from 0.5 seconds to 446.6 seconds, while the average PD decreases from 7.57% to 2.77%. From strategy 1 to strategy 2, more time is used, though the solution quality has decreased. It is because, compared to strategy 1, strategy 2 has less iterations in each temperature stage. The figure illustrates that if the cooling strategy is slower or there are more iteration at each temperature, the solution quality improves.

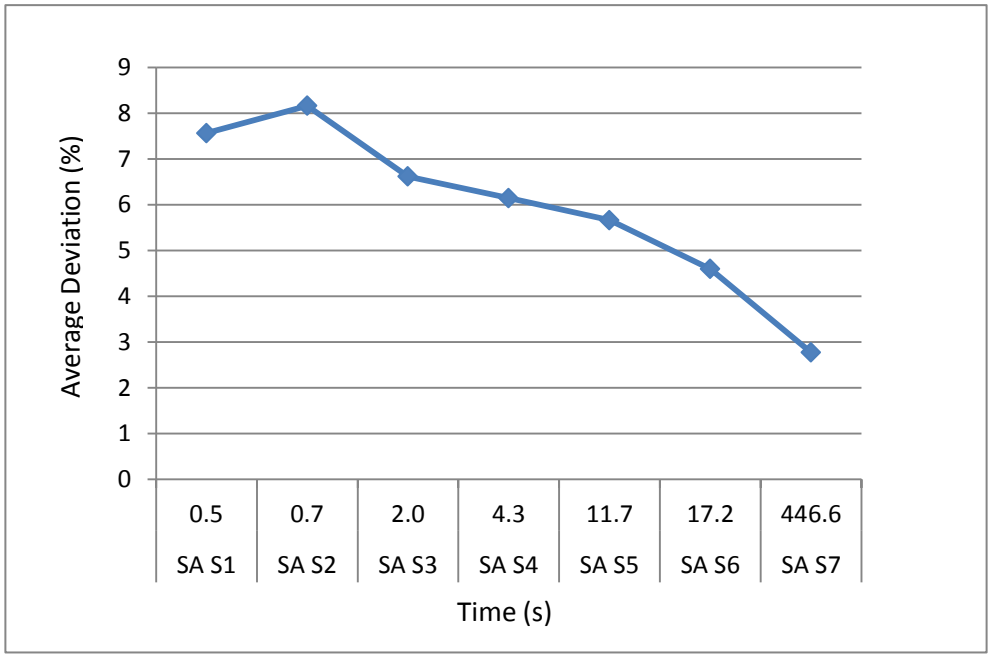

Figure 6. Results of SA analysis

5.4 Evaluation of LNS

In this section we present the computational results from the implementation of LNS. Within all tested instances, the minimum size contains 14 inbound and outbound trucks. So degrees of destruction are chosen from 2 to 14 trucks for all instances. Only first 6

cooling strategies in Table 6 are used for LNS because the computational study shows from strategy 5 to strategy 6 the improvement is minor but CPU time increases a lot.

Table 8 shows the results of LNS, where the PD is measured by $\frac{C(LNS)-C(OPT)}{C(LNS)} \times 100\ \%$, where *C(LNS*) is the best result from LNS and *C(OPT)* is the best known solution.

Truck Information			LNS1		LNS2		LNS3		LNS4		LNS5		LNS6	
In Truck	Out Truck	Product	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)	PD (%)	Time(s)
1st set														
30	32	30	1.75	113	1.75	156	1.75	447	1.75	949	0	2606	1.75	3763
32	24	32	1.96	101	1.96	140	1.96	401	1.96	867	1.96	2353	1.96	3471
32	26	32	1.89	105	1.89	146	1.89	419	1.89	905	0	2454	0	3613
32	28	32	1.82	109	0	151	0	435	0	933	0	2537	0	3762
32	30	32	1.75	113	1.75	156	1.75	447	1.75	971	0	2628	0	3879
32	32	32	32.2	117	32.2	161	32.2	462	32.2	1002	32.2	2705	32.2	4003
3rd set														
30	32	30	5.41	112	5.41	161	5.41	466	5.41	987	2.78	2674	5.41	3915
32	24	32	2.86	110	2.86	152	2.86	436	2.86	947	0	2571	0	3774
32	26	32	2.78	113	2.78	156	2.78	447	2.78	967	2.78	2623	0	3863
32	28	32	5.41	115	5.41	160	5.41	458	5.41	993	5.41	2721	5.41	3935
32	30	32	2.63	118	5.13	162	2.63	467	2.63	1012	2.63	2747	0	4102
32	32	32	2.56	127	2.56	176	2.56	505	0	1113	0	2977	0	4347
Average			**3.52**	**96**	**3.55**	**133**	**2.85**	**381**	**2.47**	**824**	**1.97**	**2235**	**1.93**	**3296**

Table 8. Results of LNS with 6 different cooling strategies

As shown in Figure 7, from the strategy 1 to strategy 6, the average CUP time increases from 96 seconds to 3296 seconds, while the average PD decreases from 3.52% to 1.93%.

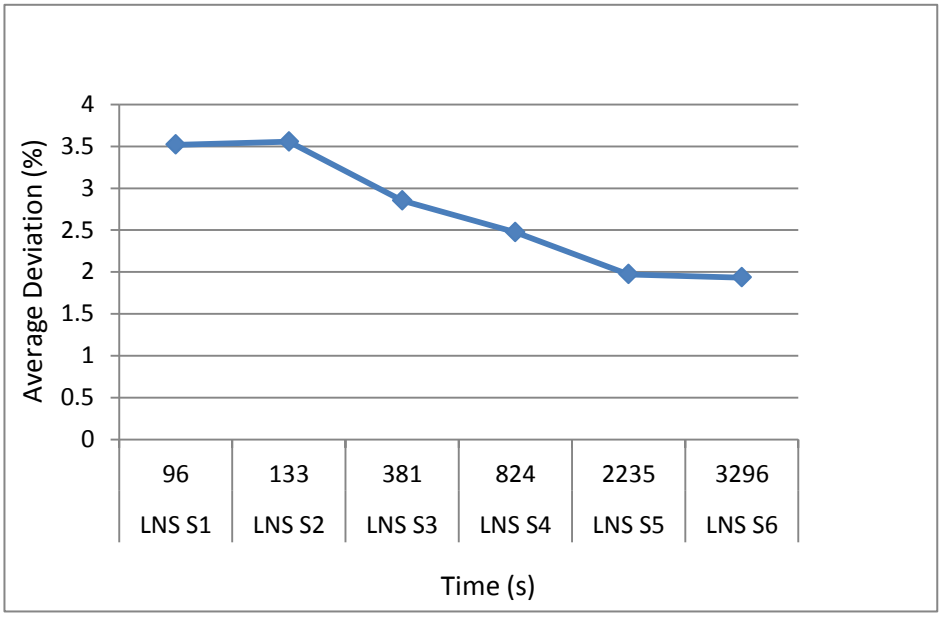

Figure 7. Results of LNS analysis

5.5 Evaluation of BS

In this section we present the computational results from the implementation of BS. As mentioned in Section 4.5, in the first phase of BS, the maximum number of nodes we can select at the first level equals to the number of inbound trucks $|I|$ and in the second phase the maximum number of nodes we can select at the first level equals to the number of outbound trucks $|O|$. But when we move downwards to the second level, the maximum number of nodes we can select is $|I|\times(|I|-1)$ and $|O|\times(|O|-1)$ for the first phase and the second phase respectively. To apply BS, depending on the different instances, we select the number of nodes at the first level to explore as $|I|$ in first phase and $|O|$ in the second phase. Thus, we then define the inbound beamwidth as $5|I|$, $10|I|$ and outbound beamwidth as $7|O|$, $9|O|$ and $10|O|$ to test all instances.

Table 9 shows the results of BS with different filtering approaches and beamwidths. The first column is the number of inbound trucks and outbound trucks. In the first row, ***BS U5 9*** means the filtering approach is based on the upper bound and inbound beamwidth and outbound beamwidth are $5|I|$ and $9|O|$ respectively. ***BS L*** means the filtering approach is based on the lower bound and ***BS G*** means the filtering approach is based on the gap between lower bound and upper bound.

Truck Inf		BS U59		BS U10 7		BS U10 12		BS L5 9		BS L10 7		BS L10 12		BS G59		BS G10 7		BS G10 12	
In Truck	Out Truck	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)	PD (%)	T (s)
1st set																			
30	32	1.75	405	1.75	644	1.75	1188	3.45	385	1.75	582	1.75	995	3.45	399	3.45	636	3.45	1040
32	24	3.85	196	3.85	323	3.85	568	3.85	158	3.85	247	3.85	418	1.96	193	1.96	322	1.96	490
32	26	3.7	247	3.7	405	3.7	731	0	211	0	330	0	558	3.7	243	3.7	398	3.7	616
32	28	3.57	313	3.57	507	3.57	950	1.82	276	1.82	427	1.82	722	3.57	311	3.57	504	3.57	794
32	30	5.08	400	5.08	642	5.08	1189	1.75	365	1.75	529	1.75	929	5.08	398	5.08	625	5.08	1025
32	32	33.3	481	33.3	800	33.3	1423	33.3	432	33.3	673	32.2	1146	33.3	478	33.3	772	33.3	1247
3rd set																			
30	32	0	402	0	664	0	1052	2.78	381	2.78	572	2.78	978	0	402	0	642	0	1042
32	24	0	200	0	343	0	506	2.86	170	2.86	259	2.86	433	0	221	0	331	0	503
32	26	2.78	257	2.78	432	2.78	648	5.41	219	5.41	329	5.41	563	2.78	252	2.78	412	2.78	642
32	28	2.78	318	0	530	0	802	5.41	278	5.41	420	5.41	713	2.78	308	2.78	501	2.78	884
32	30	0	423	0	647	0	997	0	360	0	539	0	917	0	383	0	618	0	993
32	32	0	486	0	817	0	1269	2.56	457	2.56	685	2.56	1162	0	499	0	791	0	1278
Average		**3.68**	**217**	**3.63**	**234**	**3.06**	**236**	**3.47**	**327**	**3.34**	**374**	**2.75**	**387**	**3.41**	**557**	**3.28**	**614**	**2.66**	**640**

Table 9. Results of LNS with different filtering approaches and beamwidths

As shown in Figure 8, with the same inbound and outbound beamwidths the filtering approach based on the upper bound performs best in terms of solution quality with little sacrificing in CPU time (e.g. 2 seconds more than BS G when beamwidths are 5|*I*| and 9|*O*|). With the same filtering approach the wider inbound and outbound beamwidths are the better solution quality would be.

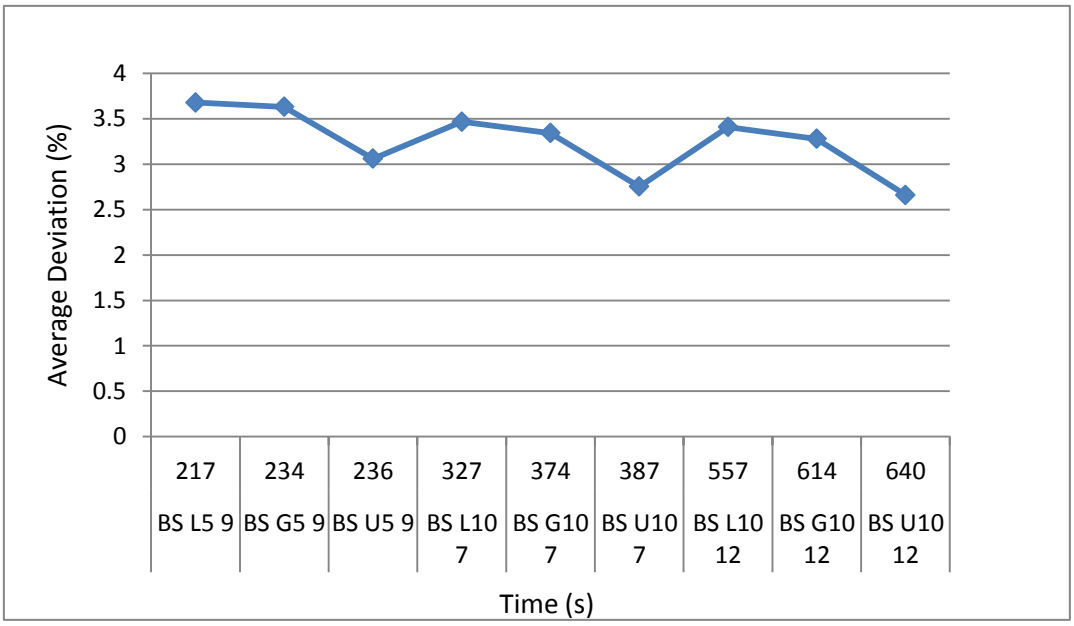

Figure 8. Results of BS analysis

5.6 Comparison of All Methodologies

The first part of this section shows the comparison of lower bounding strategies proposed by Boysen et al. [11] and the lower bounds obtained from CPLEX. In the second part, all methodologies are compared together with CPLEX. Since there are several different strategies for SA, LNS and BS, we select the strategy that provides good solutions within

reasonable time as the best strategy for each algorithm. The computational study shows that the proposed algorithms perform differently according to different structures of instances. For that reason, the algorithms are compared based on two set of instances (i.e. the first set of large instances and the third set of large instances).

5.6.1 Lower Bound Comparison

As proved in Boysen et al. [11], fixing some truck sequences leads to tighter bounds. To obtain better lower bounds, we use a similar idea as BS but here we branch every node at each level. Depending on the level, first several positions of trucks are fixed for every node thus every node at a level provides a lower bound for that partially fixed truck sequence. The minimum bound of all nodes at a certain level gives the lower bound for the whole problem. The deeper the level is the tighter the lower bound would be. However, the number of nodes goes exponentially as the level becomes deeper, e.g. the number of nodes in the 9^{th} level of an instance of the size 32 inbound trucks is 2.3×10^{14}. Due to the limited memory of our computer, we can only branch nodes to the 4^{th} level for all instances. Figure 9 shows the comparison of proposed LB and the lower bound obtained by CPLEX, where the PD is measured by $\frac{C(OPT)-C(LB)}{C(OPT)} \times 100\ \%$, where *C(LB*) is the LB and *C(OPT)* is the best known solution. The proposed LB strategies outperform CPLEX not only in terms of LB quality but also using way much less CPU time. However, compared to LB one, the LB two does not substantially improve the bounds

mainly because we only branch to the 4th level.

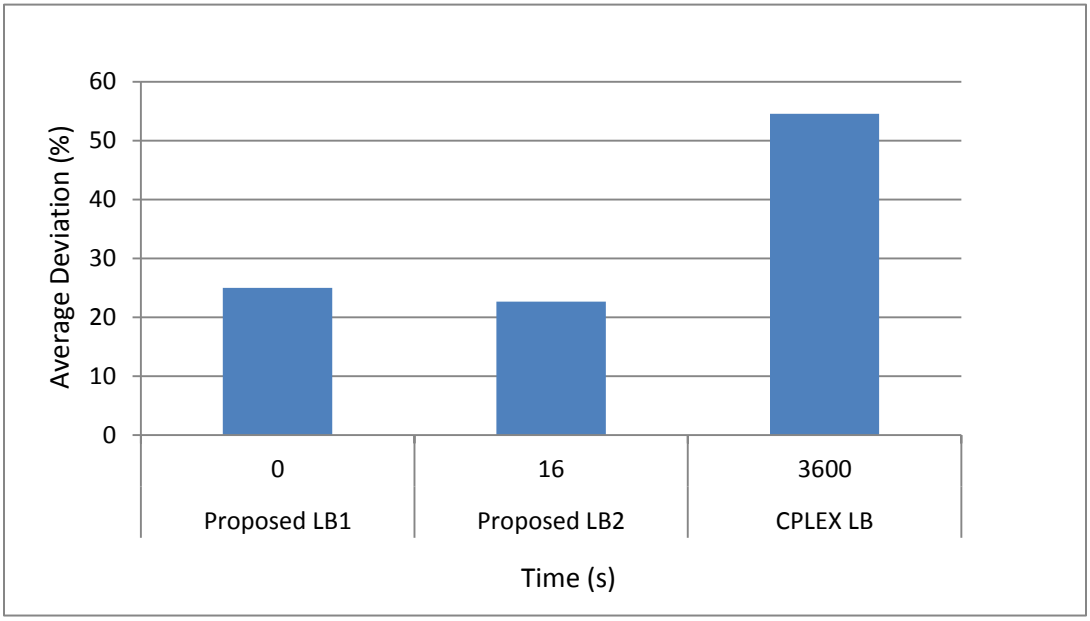

Figure 9. Results of lower bounds comparison

5.6.2 A Comparison of Heuristics with Two Sets of Large Instances

The criterion to select the best strategy for each algorithm takes in consideration both the CPU time and solution quality. The maximum CPU time of 7 strategies of SA is within several minutes (i.e. strategy 7 uses 7 minutes). With few more minutes, the solution quality of strategy 7 has obvious improvement as compared to strategy 6 so that the strategy 7 is selected. However, for LNS, the strategy 6 uses dozen minutes more than other strategies but the improvement is insignificant. For that reason, we select a strategy from 1 to 5 depending on the different structures of instances. For BS, CUP times do not have big difference among different filtering approaches when the beamwith is the same

so we select the strategy depending on the performance of each filtering approach.

For the first set of large instances, the best strategies for SA, LNS and BS are strategy 7, strategy 4 and upper bound filtering approach with 5 and 9 inbound and outbound beamwidths, respectively. As shown in Figure 10, three of our proposed algorithms perform better than CPLEX in solution quality with less CPU time. Although LS does not have better solution quality, it uses much less CPU time.

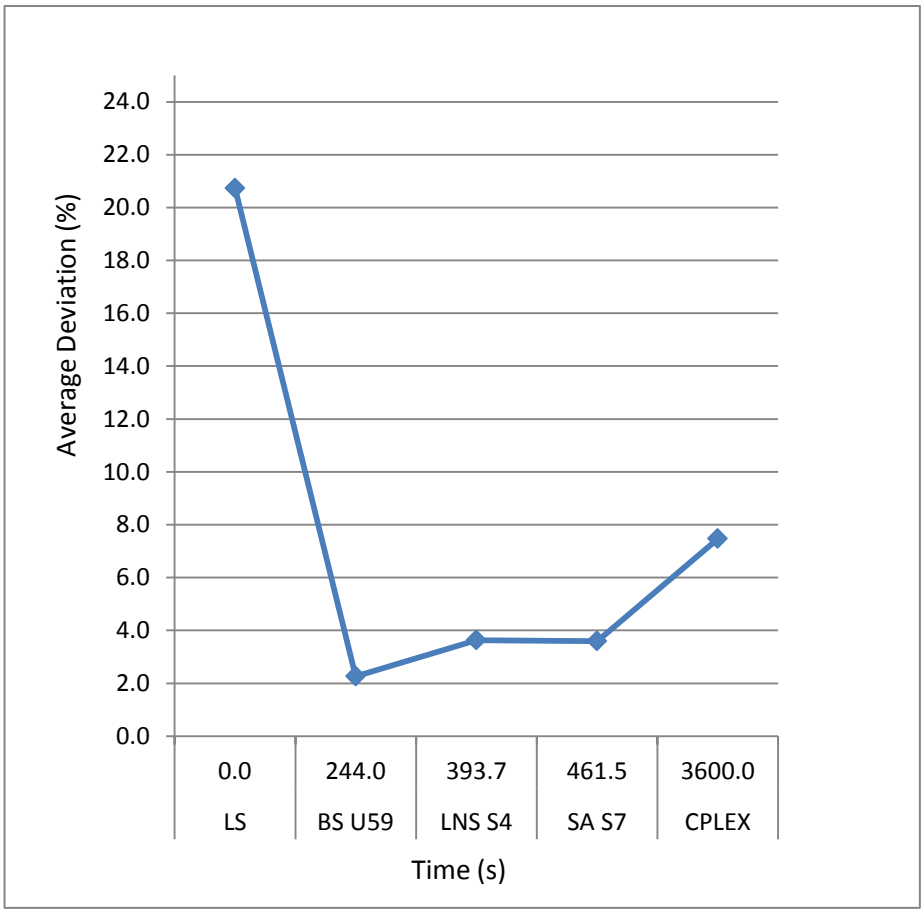

Figure 10. Comparisons of methodologies for 1st set of large instances

For the third set of large instances, best strategies for SA, LNS and BS are strategy seven, strategy 4 and lower bound filtering approach with 5 and 9 inbound and outbound beamwidth, respectively. As shown in Figure 11, CPLEX performs better in terms of solution quality with 1.4% average PD. However, our proposed algorithms use less time with reasonable solution quality, i.e. 3.8% for SA, 4.8% for BS and 4.0% for LNS.

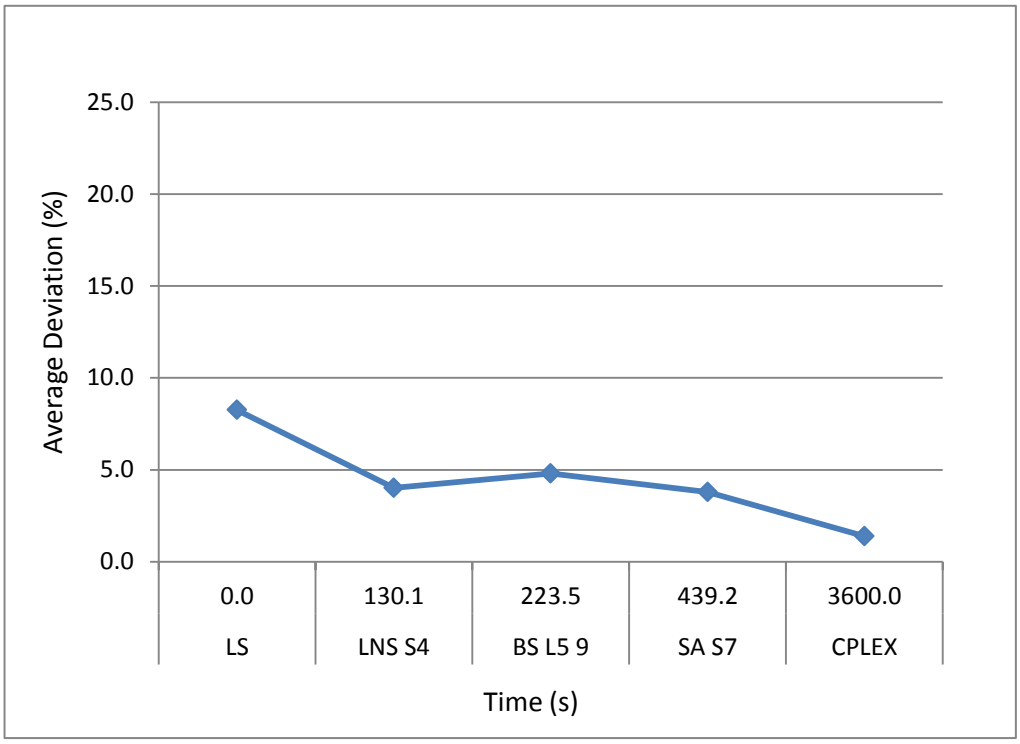

Figure 11. Comparisons of methodologies for 2nd set of large instances

Chapter 6: Conclusions and Further Research

In this thesis, we study a *trucking scheduling problem* arising in the operation of cross-docking terminals. As the most important contribution of this thesis, we develop four heuristic algorithms (local search, simulated annealing, large neighborhood search, and beam search) to deal with difficult sets of instances associated with flow structures arising in real applications. We study different strategies of each algorithm and conclude the best strategy for each one of them. In general, LNS and BS perform the best and, depending on different structures of instances, the performances of these two algorithms vary. Compared to the algorithms proposed by Boysen et al. [11], we are able to solve larger size instances with good solution qualities in reasonable CPU times. Moreover, for certain instances the proposed algorithms perform better than CPLEX, not only in terms of solution quality but also in terms of CPU times. We also test lower bound strategies that obtain tighter bounds with less time than CPLEX. In real applications, decision makers can choose one of the proposed algorithms, or CPLEX, or the hybrid of proposed algorithms and CPLEX to solve the problem according to the time constrains.

Another contribution of this thesis is to analyze the complexity of three different structures of instances. The second set of instances is the easiest and the third set of instance goes in the middle while the first set of instances is the most difficult. We also propose an alternative integer programming formulation (F2) for the problem to obtain

better linear programming relaxation bounds as compared to the formulation (F1) proposed by Boysen et al. [11]. Unfortunately, the required CPU times to solve this formulation is much higher due to the large number of variables and constrains. However, we believe that, with some decomposition technics, it would be possible to handle our formulation and to solve the problem efficiently.

We study the problem based on several assumptions and in order to solve more realistic problems, there are several aspects of this research topic that are worth further investigating.

1. Use decomposition technics to handle the proposed formulation.
2. Develop an exact algorithm. All the proposed algorithms cannot guarantee the optimal solution. Although, CPLEX is able to obtain the optimal solution for small size instances, it fails to do so for large instances. B&B based algorithms are known to be successful for optimally solving some fundamental machine scheduling problems (MSPs) [65]. The TRSP has many similarities with MSPs so that an interesting research direction would be to develop a B&B algorithm for the TRSP.
3. Consider multiple strip and stack doors. Make the original problem more realistic by taking multiple strip and stack doors into consideration.
4. Take into consideration a dynamic case. In our model, we assume the trucks are available at the beginning of the planning horizon. However, due to the traffic

congestion or other contingencies, inbound trucks may not arrive on time. Similarly, shipments are usually bound to the due dates negotiated with customers. So the dynamic approach would make the problem more applicable to the real world.

References

[1] State of Logistics. (2014). Retrieved 02/01, 2014, from http://www.ic.gc.ca.

[2] Transport Canada. (2014). Retrieved 02/01, 2014, from http://www.tc.gc.ca.

[3] Apte, U. M., & Viswanathan, S. (2000). Effective cross docking for improving distribution efficiencies. *International Journal of Logistics, 3*(3), 291-302.

[4] Boysen, N., & Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega, 38*(6), 413-422.

[5] 2008 cross-docking trends report. (2008). Retrieved 02/01, 2014, from http://www.saddlecrk.com/ Whitepaper.

[6] 2011 cross-docking trends report. (2011). Retrieved 02/01, 2014, from http://www.saddlecrk.com/ Whitepaper.

[7] Bartholdi, J. J., & Gue, K. R. (2004). The best shape for a cross-dock. *Transportation Science, 38*(2), 235-244.

[8] Galbreth, M. R., Hill, J. A., & Handley, S. (2008). An investigation of the value of cross‐docking for supply chain management. *Journal of Business Logistics, 29*(1), 225-239.

[9] Li, Y., Lim, A., & Rodrigues, B. (2004). Cross-docking—JIT scheduling with time windows. *Journal of the Operational Research Society, 55*(12), 1342-1351.

[10] Schaffer B. (1997). Implementing a successful cross-docking operation. *IIE Solutions, 29*(10), 34-36.

[11] Boysen, N., Fliedner, M., & Scholl, A. (2010). Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum, 32*(1), 135-161.

[12] McWilliams, D. L., Stanfield, P. M., & Geiger, C. D. (2005). The parcel hub scheduling problem: A simulation-based solution approach. *Computers & Industrial Engineering, 49*(3), 393-412.

[13] Yu, W., & Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research, 184*(1), 377-396.

[14] Kinnear, E. (1997). Is there any magic in cross-docking? *Supply Chain Management: An International Journal, 2*(2), 49-52.

[15] Glossary of the Material Handling Industry of America. (2011). Retrieved 02/01, 2014, from http://www.mhia.org/learning/glossary.

[16] Van Belle, J., Valckenaers, P., & Cattrysse, D. (2012). Cross-docking: State of the art.

Omega, 40(6), 827-846.

[17] Arnaout, G., Rodriguez-Velasquez, E., Rabadi, G., & Musa, R. (2010). Modeling cross-docking operations using discrete event simulation. *Proceedings of the 6th International Workshop on Enterprise & Organizational Modeling and Simulation,* pp. 113-120.

[18] Stalk, G., Evans, P., & Sgulman, L. E. (1992). *Competing on capabilities: The new rules of corporate strategy*. Harvard Business Review.

[19] Chen, F., & Lee, C. (2009). Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research, 193*(1), 59-72.

[20] Werners, B., & Wülfing, T. (2010). Robust optimization of internal transports at a parcel sorting center operated by deutsche post world net. *European Journal of Operational Research, 201*(2), 419-426.

[21] Cook, R. L., Gibson, B. J., & MacCurdy, D. (2005). A lean approach to cross docking.

[22] Napolitano, M. (2011). Cross dock fuels growth at dots. *Logistics Management (Highlands Ranch, Colo.: 2002), 50*(2).

[23] Witt CE. (1998). Cross-docking: Concepts demand choice. *Material Handling*

Engineering, 53(7), 44-49.

[24] Kreng, V. B., & Chen, F. (2008). The benefits of a cross-docking delivery strategy: A supply chain collaboration approach. *Production Planning and Control, 19*(3), 229-241.

[25] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*(4), 408-416.

[26] Agustina D, Lee CKM, Piplani R. (2010). A review: Mathematical models for cross dock planning. *International Journal of Engineering Business Management, 2*(2), 47-54.

[27] Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications, 45*(1), 41-51.

[28] Sung, C., & Song, S. (2003). Integrated service network design for a cross-docking supply chain network. *Journal of the Operational Research Society, 54*(12), 1283-1295.

[29] Gümüş, M., & Bookbinder, J. H. (2004). Cross-docking and its implications in location-distribution systems. *Journal of Business Logistics, 25(2),* 199-228.

[30] Donaldson, H., Johnson, E. L., Ratliff, H. D., & Zhang, M. (1998). Schedule-driven cross-docking networks. *Georgia Tech Tli Report, the Logistics Institute, Georgia Tech.*

[31] Musa, R., Arnaout, J., & Jung, H. (2010). Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. *Computers & Industrial Engineering, 59*(1), 85-92.

[32] Sung, C., & Yang, W. (2008). An exact algorithm for a cross-docking supply chain network design problem. *Journal of the Operational Research Society, 59*(1), 119-136.

[33] Bartholdi III, J. J., & Gue, K. R. (2000). Reducing labor costs in an LTL crossdocking terminal. *Operations Research, 48*(6), 823-832.

[34] Kirkpatrick, S., Gelatt, C. D.,Jr, & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science (New York, N.Y.), 220*(4598), 671-680.

[35] Lim, A., Miao, Z., Rodrigues, B., & Xu, Z. (2005). Transshipment through crossdocks with inventory and time windows. *Naval Research Logistics (NRL), 52*(8), 724-733.

[36] Chen, P., Guo, Y., Lim, A., & Rodrigues, B. (2006). Multiple crossdocks with inventory and time windows. *Computers & Operations Research, 33*(1), 43-63.

[37] Glover, F., & Kochenberger, G. A. (2003). *Handbook of metaheuristics* Springer.

[38] Lee, Y. H., Jung, J. W., & Lee, K. M. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers & Industrial Engineering, 51*(2), 247-256.

[39] Liao, C., Lin, Y., & Shih, S. C. (2010). Vehicle routing with cross-docking in the

supply chain. *Expert Systems with Applications, 37*(10), 6868-6873.

[40] Wen, M., Larsen, J., Clausen, J., Cordeau, J., & Laporte, G. (2009). Vehicle routing with cross-docking. *Journal of the Operational Research Society, 60*(12), 1708-1718.

[41] Peck, K. E. (1983). *Operational Analysis of Freight Terminals Handling Less than Container Load Shipments,* Ph.D. thesis, University of Illinois at Urbana-Champaign.

[42] Tsui, L. Y., & Chang, C. (1990). A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering, 19*(1), 309-312.

[43] Bermúdez, R., & Cole, M. H. (2001). *A Genetic Algorithm Approach to Door Assignments in Breakbulk Terminals,* Technical report MBTC 1084, Fayettevilee, AR, USA: Mack-Blackwell Rural Transportation Center, University of Arkansas.

[44] Bozer, Y. A., & Carlo, H. J. (2008). Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions, 40*(11), 1007-1018.

[45] Bartholdi III, J. J., & Gue, K. R. (2000). Reducing labor costs in an LTL crossdocking terminal. *Operations Research, 48*(6), 823-832.

[46] Yu, V. F., Sharma, D., & Murty, K. G. (2008). Door allocations to origins and destinations at less-than-truckload trucking terminals. *Journal of Industrial and Systems*

Engineering, 2(1), 1-15.

[47] Yu, W. (2002). *Operational Strategies for Cross Docking Systems,* Iowa State University, Ph.D. Dissertation

[48] Vahdani, B., & Zandieh, M. (2010). Scheduling trucks in cross-docking systems: Robust meta-heuristics. *Computers & Industrial Engineering, 58*(1), 12-24.

[49] Boloori Arabani, A., Fatemi Ghomi, S., & Zandieh, M. (2011). Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert Systems with Applications, 38*(3), 1964-1979.

[50] Forouharfard, S., & Zandieh, M. (2010). An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems. *The International Journal of Advanced Manufacturing Technology, 51*(9-12), 1179-1193.

[51] Michiels, W., Aarts, E., & Korst, J. (2007). *Theoretical aspects of local search* Springer.

[52] Vahdani, B., Soltani, R., & Zandieh, M. (2010). Scheduling the truck holdover recurrent dock cross-dock problem using robust meta-heuristics. *The International Journal of Advanced Manufacturing Technology, 46*(5-8), 769-783.

[53] Soltani, R., & Sadjadi, S. J. (2010). Scheduling trucks in cross-docking systems: A

robust meta-heuristics approach. *Transportation Research Part E: Logistics and Transportation Review, 46*(5), 650-666.

[54] Larbi, R., Alpan, G., Baptiste, P., & Penz, B. (2011). Scheduling cross docking operations under full, partial and no information on inbound arrivals. *Computers & Operations Research, 38*(6), 889-900.

[55] Alpan, G., Larbi, R., & Penz, B. (2011). A bounded dynamic programming approach to schedule operations in a cross docking platform. *Computers & Industrial Engineering, 60*(3), 385-396.

[56] Rosales, C. R., Fry, M. J., & Radhakrishnan, R. (2009). Transfreight reduces costs and balances workload at georgetown cross-dock. *Interfaces, 39*(4), 316-328.

[57] Wang, J., & Regan, A. (2008). Real-time trailer scheduling for crossdock operations. *Transportation Journal,* 47(2) , 5-20.

[58] Acar, M. K. (2004). *Robust Dock Assignments at Less-than-Truckload Terminals,* Master's thesis, University of South Florida.

[59] Lowerre, B. T. (1976). The HARPY speech recognition system. Ph.D. thesis, Carnegie-Mellon University.

[60] Lim, A., Ma, H., & Miao, Z. (2006). Truck dock assignment problem with time

windows and capacity constraint in transshipment network through cross-docks. *Computational science and its applications-ICCSA 2006* (pp. 688-697) Springer.

[61] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Principles and practice of constraint Programming—CP98* (pp. 417-431) Springer.

[62] Lim, A., Ma, H., & Miao, Z. (2006). Truck dock assignment problem with time windows and capacity constraint in transshipment network through cross-docks. *Computational science and its applications-ICCSA 2006* (pp. 688-697) Springer.

[63] Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research, 37*(1), 32-41.

[64] Kuo, Y. (2013). Optimizing truck sequencing and truck dock assignment in a cross docking system. *Expert Systems with Applications, 40*(14), 5532-5541.

[65] Pinedo, M. (2008). *Scheduling* (3rd ed.). New York; London: Springer.