Vehicle Routing with Cross Docks, Split Deliveries, and Multiple Use of Vehicles

by

Arun Kumar Ranganathan Jagannathan

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 12, 2011

Keywords: VRP, Cross dock, Integer programming

Approved by

Abstract

Cross docking plays a vital role in supply chain management. Cross docks can reduce the lead time of the product from supplier to retailer. The benefit of the cross dock would be reduced without efficient vehicle routing and scheduling. This research work proposes a mixed integer linear programming formulation to obtain feasible vehicle routes and schedules. The vehicle routing problem is characterized by heterogeneous vehicles, split deliveries, discrete time windows, linehaul, and backhaul operations. The problem is modeled to facilitate the pick up of empty pallets at the retailer during the linehaul operation and delivery of empty pallets to the suppliers during the backhaul operation. The mathematical formulation is solved using CPLEX. The research work also proposes an algorithm to obtain initial feasible vehicle routes.

Acknowledgments

I would like to thank Dr Chase C. Murray, for his guidance and for providing me an opportunity to work on my research work. I would like to thank Dr Jeffrey S. Smith and Dr Kevin R. Gue for their reviews on my thesis work and for serving on my committee.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today in many industries, logistics is one of the major cost factors. For example, Bayer AG, a chemical company with annual sales equivalent to \$25 billion, has a logistics budget of \$5 billion; it involves 3,000 different distribution points with handling about 740,000 different shipments (Johnson and Wood 1990). Trucking accounts for 83% of freight transportation in the US alone (Wilson 2002). A continuous surge in oil prices and increasing competition have forced industries to implement novel cost-reduction strategies.

Distribution centers are important components of the logistics distribution network. They facilitate the consolidation of goods from a variety of suppliers to a variety of retailers. According to the functions of the distribution center, there is a distinction between an inventory coordination point and an inventory storage point. The former is a cross docking strategy, while the latter is a traditional warehousing strategy (Kreng and Chen 2008). The cross docking strategy has been acknowledged as having great potential to reduce transportation costs and delivery times without increasing inventory (Sung and Song 2003).

Two critical requirements of cross docking are simultaneous arrival and consolidation. If all vehicles do not arrive at the cross dock simultaneously, some vehicles have to wait. Therefore, the timing for all vehicles in the pickup process must be synchronized to reduce waiting times. According to the destination, all products are classified and loaded to each vehicle in the cross dock. Then, all vehicles leave the cross dock to distribute products to their destinations. If simultaneous arrival and consolidation can be easily accomplished in a supply chain's physical flow, all products can be moved from suppliers to customers without any interruptions. Therefore, we can expect to reduce inventory levels and lead times for delivery (Lee et al. 2006). In the case of Wal-Mart, cross docking is often regarded as a key driver of the retailer's superior logistics management (Hammer 2004).

The benefits of cross docks do not come without some risks. For example, Apte and Viswanathan (2000) caution that, "cross docking inherently leads to a minimal level of inventory at the warehouse, and thereby strips the system of safety stocks traditionally held at the warehouse. Consequently, cross docking raises the probability of stock-out situations." Fortunately, this issue may be addressed by utilizing the concept of vendor managed inventory (VMI). With VMI, the supplier (which is often a manufacturer) controls the buyer's (retailer's) inventory levels, so as to ensure that predetermined customer service levels are maintained. In such a relationship, the supplier takes the replenishment decision for the buyer, dispatching a quantity of product that may be fixed (Waller et al. 2006). Replenishment occurs when the stock level at the buyer reaches a specified level, based on both the average demand during transportation lead time and a safety stock to cover for demand variations (Kaipia et al. 2002). VMI brings a number of benefits to all parties participating in the supply chain. First, the impact of demand amplification is dampened, as the manufacturer now receives a direct view of end consumer demand patterns and can use this in forecasting (Disney et al. 2003). This generates cost benefits through a reduction in buffer stocks at the buyer and supplier (Sabath 1995). Further, there are improvements in service levels as product availability is increased (Waller et al. 2006). Finally, VMI can, in the long run, increase the profitability of both the supplier and buyer in supply chain (Dong and Xu 2002). The buyer benefits from lower inventory costs and can offer a price reduction. This then increases sales volume, which benefits both parties in terms of increased profitability (Disney et al. 2003).

Motivated by the usefulness of the cross docks, several authors have focused on improving the performance within the cross dock itself. Some of the notable studies include the design of the cross dock layout (Bartholdi and Gue 2000; Bartholdi and Gue 2004), benefits of the cross dock layout when compared to traditional warehouse (Kreng and Chen

2008), analysis of retailer inventory levels when cross docks are used (Waller et al. 2006) and scheduling of trucks at cross dock doors (Boysen and Fliedner 2010). However, less work has been done on the problem of routing and scheduling in a cross dock distribution network.

The distribution network has several key components. These include suppliers, consumers (retailers), cross docks, and the vehicles that transport goods between nodes in the network. It is important to select the proper mix of vehicles, and to schedule them appropriately. The vehicles may be routed in a truckload operation or a less-than-truckload operation. Less-than-truckload operations are observed if the loaded capacity of the vehicle for a particular route will be less than the capacity of the vehicle. Transportation costs may increase if less-than-truckload operations are frequently scheduled. If the customers are willing to receive goods in multiple shipments, a practice known as "split delivery", the occurrence of less-than-truckload operations may be reduced. The operations of the truck may have only linehaul operations or it may have both linehaul and backhaul operations. In linehaul operations, the goods are either picked up or delivered. In backhaul operations, after goods are delivered to customers, trucks are routed for pick up from the suppliers. The combination of the linehaul and the backhaul operation is considered efficient, as it may reduce the occurrence of vehicles travelling empty (dead heading).

This research effort makes several contributions. A mathematical model is developed for the vehicle routing problem (VRP) for cross docks. Some work has been reported on the VRP for cross docks, with extensions like time windows, split deliveries, heterogeneous fleet of vehicles, and multiple cross docks. All the reported work has one of the above mentioned extensions, or a combination of only a couple of extensions to the VRP for cross docks. In this thesis, the proposed VRP model is built with all the above mentioned extensions, in addition to three new extensions. First, to improve the vehicle utilization, a feature to allow pick up during a linehaul operation is added. Second, backhaul operations are added. Finally, a feature to allow delivery during a backhaul operation is added. In addition to developing a mathematical model, a heuristic approach is proposed for determining feasible vehicle routes.

The solution from the heuristic approach can be used within a broader solution approach, like tabu search, to obtain better solutions. Moreover, a procedure is explained to develop test problems for this model.

The remainder of this thesis is organized as follows. A formal description of the distribution problem involving cross docks, backhaul operations, and multiple use of vehicles is presented in Chapter 2. Chapter 3 contains a review of relevant literature. In Chapter 4, parameters and decision variables used in the development of the mixed integer linear programming formulation for the model are explained. Moreover, network constraints, cross dock constraints, supplier pick up constraints and retailer delivery constraints of the model are also explained. In Chapter 5, a numerical example is provided to further explain the details of the mathematical model. A heuristic approach for obtaining initial feasible vehicle routes is explained in Chapter 6. Numerical analysis of the mathematical model for different parameter values is included in Chapter 7. Finally, concluding remarks and suggestions for future research are given in Chapter 8.

Chapter 2

Problem Description

The problem of interest involves a distribution network consisting of retailers, suppliers, cross docks, and vehicles of potentially-differing capacities. Goods must be delivered from suppliers to retailers through one or more cross docks. Within each cross dock, goods arriving on multiple inbound trucks are sorted and consolidated onto multiple outbound trucks. To facilitate efficient flow of goods through the cross dock, orders are assumed to be loaded onto pallets of uniform size.

Each retailer has a known demand of goods from each supplier. The allowable times in which deliveries may be made to each retailer are defined by one or more time windows, such that deliveries may not be scheduled outside of these windows. Split deliveries, in which multiple vehicles may deliver goods at separate times, are allowed in this system in an effort to maximize vehicle utilization. The delivery of goods to retailers is described by a linehaul operation. As goods are delivered to retailers, an inventory of empty pallets may accumulate. These empty pallets must be returned to the appropriate suppliers as part of a backhaul operation.

As with retailers, vehicles may pick up goods from suppliers only within pre-defined time windows. These time windows also define the allowable times in which vehicles may return empty pallets, which were delivered as full pallets to retailers as part of the linehaul operation, to a given supplier. Split deliveries of empty pallets are permissible, and may be necessary due to vehicle capacity limitations, time window constraints, or vehicle coordination issues at the cross dock.

The actual quantity of each retailer's demand is assumed to be determined by the application of vendor managed inventory (VMI). As such, the demand of each retailer is known, and there is no need to maintain inventory levels in the distribution centers (in this case, the cross docks). By assumption, there is no space to store inventory in the cross dock. In order to operate an inventory-free cross dock, the transfer operation within cross docks requires the applicable vehicles to arrive at the cross dock simultaneously. Goods are assumed to travel directly from inbound to outbound trucks.

There may be multiple cross docks in pre-defined (fixed) locations. Each cross dock consists of a set of terminals for inbound trucks (containing pallets from suppliers) and another set of terminals for outbound trucks (to be loaded with pallets for delivery to retailers). It is important to note that the problem of interest here is not concerned with the scheduling of vehicles to particular terminals in the cross dock. Therefore, it is not practical to consider terminal-specific transfer times between incoming and outgoing shipments.

A heterogeneous fleet of vehicles (trucks) is available, such that each vehicle may have a unique capacity. Vehicles may be assigned to multiple routes over the course of the planning horizon. Each truck is assumed to begin empty at its current location at the beginning of the planning horizon. Trucks may travel from their initial locations to either a supplier (where the pickup process for goods commences) or to the outbound terminal of a cross dock (where trucks wait to be loaded with goods picked up by other trucks visiting suppliers). Less-than-truckload (LTL) operations are allowed. Because all goods bound for retailers must pass through a cross dock, vehicles may not travel directly from suppliers to retailers. However, vehicles may deliver empty pallets directly from retailers to suppliers as part of a backhaul operation.

The objective of the problem is to determine the minimum-cost vehicle schedules, where the total cost may be calculated in multiple ways. For example, there may be a cost associated with the number of vehicles utilized. To facilitate more equitable numbers of routes for each vehicle, a cost may be included for each route taken by each vehicle. Additionally, a preference may be given for delivering to retailers as early as possible within their allowable time windows. Although VMI is used to establish demand values, it is possible that not all demand can be satisfied. This may be due to vehicle capacity limitations, cross dock availability issues, or time window limitations. As such, another cost may be associated with the quantity of unmet demand, both for goods needed by retailers and for empty pallets requested by suppliers.

Chapter 3

Literature Review

The problem of interest here may be considered as a vehicle routing problem (VRP) with multiple side constraints. Such constraints include, but are not limited to, split deliveries, time windows, and heterogeneous vehicles in a cross docking environment. Many studies have been done separately on the VRP with varying constraints, and on cross docking. However, very little research exists on the integration of the VRP with cross docking. This literature review is divided into two parts. First, literature describing the VRP with cross docks is considered. Next, relevant vehicle routing problems in non-cross dock environments are reviewed.

3.1 VRP and Cross Docking

This section contains a review of articles involving vehicle routing, sequencing, and scheduling of trucks in a cross docking environment. A summary of the literature appears in Table 3.1, which also contrasts aspects of the existing literature with the model proposed in this thesis. In Table 3.1, the “Backhaul Operation” column indicates whether vehicles are permitted to visit suppliers after making delivery to retailers. The “Pick up at Linehaul” column indicates whether vehicles may pick up empty pallets during the time of delivery of full pallets to retailers.

Author	Multiple Cross Dock	Split Delivery	Time Windows	Hetero. Vehicle	Backhaul Operation	Pick up at Linehaul
Sung and Song (2003)	Yes	No	No	Yes	No	No
Gumus and Bookbinder (2004)	Yes	Yes	No	No	No	No
Chen et al. (2006)	Yes	No	Yes	No	No	No
Lee et al. (2006)	No	No	No	No	No	No
Wen et al. (2008)	No	No	Yes	No	No	No
Liao et al. (2010)	No	No	No	No	No	No
Miao et al. (2010)	Yes	No	Yes	No	No	No
Musa et al. (2010)	Yes	No	No	No	No	No
Soltani and Sadjadi (2010)	No	No	No	No	No	No
Ma et al. (2011)	Yes	Yes	Yes	No	No	No
This Thesis	Yes	Yes	Yes	Yes	Yes	Yes

Table 3.1: A comparison of cross dock models

Sung and Song (2003) develop an integrated service network design problem based on a path based formulation. The authors have chosen a path based formulation approach instead of an arc-flow based formulation approach on the possibility of getting an efficient and effective formulation. The authors give a predefined cut off time at the origin nodes within which all packages should be loaded and a predefined critical entry time at the destination nodes after which all the packages should be delivered. The consolidation of the packages should take place between those times. The problem is modeled considering multiple cross docks and multiple origin and destination nodes. The problem is solved using a tabu search heuristic, and a separation heuristic was developed for identifying violated valid inequalities.

Gumus and Bookbinder (2004) studied cross docking and its implications in the location distribution system. A model was developed to determine the best locations to operate the cross docks. The model was built taking into consideration transportation costs, facility costs, inventory costs at the manufacturers and retailers, and in-transit inventory costs. The research work also addresses the issue of the location for product consolidation.

Lim et al. (2005) developed transshipment models with time window constraints and addressed the issue of the just-in-time objective to the cross docks. The time windows considered in this work include both fixed and flexible time windows. In a fixed time window, shipping and delivery at suppliers and customers are allowed only at a particular time. In a flexible time window, shipping and delivery are allowed at any time within the time window. The authors also used mixed time windows, a mixture of fixed and flexible time windows. The aim of this research is to minimize inventory delay and to find optimal schedule for trucks in the transshipment centers.

Chen et al. (2006) proposed a cross docking problem to minimize the total cost in the distribution plan within the specified time windows. In this model, capacity constraints were considered at the cross docks and an inventory holding cost was assigned for the amount of inventory held at the cross dock. The initial solution to this problem was obtained by a greedy method. The authors used simulated annealing, tabu search, and a hybrid heuristic method to solve the problem.

Lee et al. (2006) developed a mathematical model and proposed a heuristic algorithm based on tabu search with an objective of minimizing the transportation cost and fixed cost of vehicles, and to determine the number of vehicles, optimal vehicle routing, and scheduling with cross docks. The assumptions of the model include homogeneous vehicles with no split deliveries, with all vehicles starting and returning to a cross dock. Time window constraints were developed only at the cross dock to facilitate simultaneous arrival of vehicles and the consolidation process, and to reduce the waiting time of vehicles. No time windows were considered for the suppliers and retailers. Separate vehicles were used for the pickup and delivery operation. The proposed algorithm first finds the initial solution, which gives the possible nodes to which each vehicle may travel from the existing position by selecting the route which has a ratio of transportation cost to minimum transportation cost less than a tolerance value α of 1.5. Then the remaining nodes are allotted randomly. The solution was improved by exchanging the nodes with corresponding vehicle routes. This algorithm was tested for up to 50 nodes. Results were compared with results obtained from the enumeration process. Effects of tolerance value α on results were analyzed.

Wen et al. (2008) developed a mixed integer programming formulation to solve the VRP with cross docking and time window constraints on the nodes with the objective of reducing the total travel time. The authors solved the problems using a tabu search heuristic embedded within an adaptive memory procedure (AMP). Instead of simultaneous arrival and departure of vehicles at the cross dock, as proposed by Lee et al. (2006), the authors determined the dependency among the vehicles based on consolidation decisions. A homogeneous set of vehicles was considered. In the consolidation process a set of vehicles pick up materials from the suppliers and deliver to the cross dock. This same set of vehicles deliver materials from the cross dock to retailers. Time windows were considered only at the nodes with no split deliveries. The authors considered a time horizon for whole transportation operation. In the AMP, the set of vehicle tours made by vehicles was stored in adaptive memory and the initial solution was formed by combining the vehicle tours. In the neighborhood search phase, the authors proposed two approaches: small neighborhood search and large neighborhood search. In small neighborhood search, the algorithm tries to find the best solution by moving the nodes from a small subset of nodes to a small subset of vehicles. If the best solution is not found, the algorithm alternates to find the best solution from the large neighborhood (i.e., from the whole solution space). Aspiration criterion was used to avoid repeated assignment of nodes to the same vehicles. The algorithm was tested for up to 200 nodes. To reduce the computational time, the authors used the aggressive skip technique, where, when a feasible route is identified for a vehicle, this skip applies. This method led to a considerable reduction in computational time when compared to the conservative skip technique.

Liao et al. (2010) extended the work of Lee et al. (2006) and proposed a heuristic algorithm based on tabu search, with the aim of reducing the computational time and the number of vehicles used. The objective of the problem was to reduce the transportation and operational cost. The initial solution was developed in two steps. In the first step, the pick up process was considered. The algorithm calculated the minimum number of vehicles required and assigned the nodes to the vehicles. The same procedure was followed in step two for the delivery process. In the neighborhood search phase, total cost was minimized by continuously arranging the nodes from one vehicle to another. In order to reduce the operational cost, only full truck load capacities were considered. The tabu search algorithm removed empty vehicles from the system, while it was disallowed in Lee et al. (2006). Moreover, this paper showed differences in the assignment of nodes to vehicles. The authors showed improved performance of their algorithm by considering the same parameters of Lee et al. (2006) and by comparing the computational time. However, the model has its own limitations. One of the limitations is that each supplier or retailer can be picked up only once and the quantity picked up must not exceed the capacity of the vehicle. If the demand of the product at a retailer from the supplier is greater than the capacity of the truck, this model cannot be used.

Miao et al. (2010) developed a VRP problem with soft and hard time windows. The hard time window is the allowable time a node can be serviced and the soft time window is the preferred time window for the node within the hard time window. The authors developed a transshipment model with cross docks to develop a transportation schedule with the aim of minimizing the transportation cost, inventory handling, and penalty costs. A penalty cost is added to the cost factor if cargo is not delivered to the customer within the soft time window. The model consists of multiple cross docks with multiple suppliers and customers. The authors employ a three stage assignment process to develop initial solutions, where stage one is the supplier cross dock route assignment, stage two is customer cross dock route assignment, and stage three is a deterministic repair strategy to adjust schedules to obtain feasible solutions. The problem was solved using two heuristic approaches, adaptive tabu search and adaptive genetic algorithm.

Musa et al. (2010) developed an integer programming formulation for solving a VRP in a cross docking environment. The objective of the article was to reduce the transportation cost by routing the vehicles directly from suppliers to retailers or through the cross dock. The authors proposed a new ant colony optimization algorithm (ACO) to solve the model. The authors used more than one cross docking facility in this problem. The authors have considered a homogeneous fleet with unlimited supply of trucks. The less than truck load is assumed for direct supply to the retailers from suppliers with no time window constraints and split deliveries. The problem did not mention the coordination of the trucks in the cross docks or the movement of goods between the cross docks. By using the ACO technique, the authors were able to determine the number of trucks required to travel directly from suppliers to retailers and the trucks that visit the cross dock(s) and to determine the transportation cost. Like most optimization algorithms, the authors embedded a local search algorithm in the ACO algorithm which searches for neighboring solutions to get the improved transportation cost. The parameters for the ACO algorithm were found by using design of experiments (DOE), a statistical method. D-optimal design was used to reduce the computational time, by reducing the number of runs at the same time to get better results. The factors in this experiment were analyzed using regression and ANOVA techniques. The computations were performed for up to 75 origin and destination points and for up to 50 cross docks. The proposed algorithm was able to outperform the branch and bound algorithm in the case of large problem instances.

Soltani and Sadjadi (2010) developed a model to optimize the scheduling of trucks in cross docking systems (i.e., to find the matching pairs of inbound and outbound trucks and sequence them). The authors used a metaheuristic approach, hybrid simulated annealing, and hybrid simulated variable neighborhood search to find the truck pairs. The model was built taking into consideration one cross dock, multiple suppliers, multiple retailers, and multiple products. The idea behind the paper is to minimize the flow time of products in the cross dock by sequencing and scheduling the transportation facilities. For larger problems the authors expected an exponential growth of variables and constraints, so the mixed integer formulation is not used. Moreover, for the mathematical models the computational time was expected to be more. The authors expected the computational complexity to increase with the increase of inbound and outbound trucks and product types when heuristics were used. Therefore, the authors used a metaheuristic approach to solve the problem. The robustness of the proposed algorithm was tested using the Taguchi scheme. The Taguchi parameter design method and ANOVA were used to find the parameters that had significant impact on the algorithm. The algorithm was tested for up to 20 inbound trucks, 20 outbound trucks and 12 product types.

Ma et al. (2011) developed a model to study the issue of shipment consolidation and transportation in a cross docking environment. The model was developed taking into consideration the transportation costs, truck setup costs, the number of trucks used, and their trade-offs. The model was formulated as an integer programming model and solved using a two-stage heuristic algorithm. In the first stage, the truck load plan was developed to determine a less-than-truckload plan. In the second stage, the supply and demand at the nodes that are not satisfied are quantified.

3.2 VRP and Non Cross Docking Environment

This section contains a review of articles of VRP in a non cross docking environment. The review contains studies on the vehicle routing problem with constraints such as time windows, split deliveries, and use of a heterogeneous fleet.

Yano et al. (1987) developed vehicle routing software with an objective of minimizing the total cost of routes and to make sure that all the transportation requirements are fulfilled. The authors proposed a set covering approach based on Lagrangian relaxation in a branch and bound framework. The constraints were framed to minimize the total number of trucks. The authors considered the pickup and delivery process by the same vehicles, while the linehaul operation is completed before the backhaul operation starts. In this problem, the total number of deliveries and pick ups were restricted to four, with no split deliveries or time window constraints. Moreover, the authors have considered the origin and destination of the trucks as the warehouse.

The authors used two heuristics to find good initial feasible solutions. The first heuristic was used to find the routes that did not completely satisfy the transportation requirements and to find an initial feasible solution. The second heuristic was used to minimize the cost difference between the initial feasible solution and the optimal solution. A branch and bound procedure was used in the optimization process. The algorithm used an efficient lower bounding technique and the nodes in the branch were characterized by unordered Cartesian products. Subgradient optimization was used to determine appropriate Lagrangian multipliers at the nodes. The database created by the authors consists of 50 stores and 200 suppliers. The algorithm was able to work effectively with problems containing 20 to 40 transport requirements only. One transportation requirement was considered as the movement of one truck from warehouse to retailers to suppliers and back to warehouse.

Toth and Vigo (1997) proposed the asymmetric version of the VRP with backhauls, where the distance between each pair of locations is not same in both directions. Symmetric versions were also considered. The objective function was to minimize the transportation cost. The authors proposed an integer linear programming model with vertices consisting of depot, linehaul, and backhaul nodes. The arc set was disjointed into three subsets (i.e., arcs from depot to linehaul vertices, backhaul vertices to depot, and linehaul vertices to backhaul vertices). Arcs connecting backhaul to linehaul vertices were not allowed.

The authors used three different relaxation techniques. The purpose of the first relaxation technique is to lead to the solution of the transportation problem by dropping the capacity cut constraints. The second relaxation was based on the projection of the feasible solution space by determining the shortest spanning tree or arborescence and optimal solution to the minimal cost flow problems. Third, a Lagrangian lower bound was derived by including the relaxed degree constraints in the objective function. The lower first branch and bound algorithm was used to find the optimal solution for the vehicle routing problem. Variable reduction and feasibility checks were used to improve the performance of the branch and bound algorithm. The authors were able to provide computational results for up to 100 customers.

Mingozzi et al. (1999) proposed a new integer programming formulation for solving the VRP with backhauls. The authors proposed a procedure to calculate the lower bound to the optimal solution by combining different heuristic methods. This procedure was used to solve the dual of the LP relaxation. The authors considered a homogeneous fleet with no constraints on time windows or split deliveries. All backhaul customers were visited before the linehaul customers. Depot, linehaul, and backhaul customers were considered as vertices on a directed graph with non-negative costs on the arcs. The authors found the lower bound for this problem by solving the dual problem of the LP relaxation using heuristic procedures. The authors used the dual solution to obtain a feasible route and then an optimal solution by reducing the number of routes. A procedure was proposed to reduce the number of variables for solving using the branch and bound method. The authors used CPLEX, an LP solver, to solve the problem for up to 100 customers.

Privé et al. (2005) considered a VRP with the distribution center for a soft drink industry with heterogeneous fleet, time windows, capacity, and volume constraints. The objective of the problem was to minimize the routing costs minus the revenue derived from the collection of empty recyclable containers. The vehicle movement is between the distribution center (DC) and the retailers with origin and destination of the vehicles at the DC. The authors did not consider the pick up process from the bottling center (suppliers) to the DC. The pick up process in this problem is the pickup of empty cans from the retailers to the DC. No split deliveries were allowed and the time horizon is one day. The problem formulation is based on graph theory, where the DC and the retailers were considered as vertices and arcs the routes. The authors solved the problem using three heuristics, one is neighborhood search and the other two are improvement procedures which use a combination of 3-opt and

2-interchange moves and route mergers. The nearest neighborhood search was used to find the set of feasible routes and finds the route and vehicle with smallest cost per customer visited. The first petal heuristic considers the problem as a set partitioning problem and second petal heuristics considers till the route expansion phase in first petal heuristics and finds an optimal selection from the possible routes based on nearest neighborhood search. The improved solution was obtained by applying 3-opt and 2-interchange techniques.

Vehicle routing problem with backhauls (VRPB) is a variant of VRP where the same vehicle performs both the linehaul and backhaul operations in a single route. Normally the backhaul operation is performed once the linehaul operation is completed. Linehaul operation is the delivery process and backhaul is the pickup process. The backhaul operation proposed in this article is relevant to the delivery of empty pallets and pickup of full pallets in the proposed thesis work.

Gulczynski et al. (2011) proposed a new variant of vehicle routing problems called the "multiple depot split delivery vehicle routing problem." A mixed integer programming based model is developed, where a set of customers is served from multiple depots, and all customers accept split deliveries. The objective is to reduce the travel distance. A comparison study is performed to determine the reduction in distance travelled, by allowing split deliveries among vehicles based at the same depot and vehicles based at different depots. A heuristic is proposed to solve the problem. First, each customer is assigned to the nearest depot based on distance. Second, a heuristic is developed with constraints such that customers will receive goods only from their assigned depot. Third, a heuristic is developed that allows customers to receive goods from multiple depots. The authors, through their numerical analysis, identified that the benefits obtained by allowing split deliveries to customers served by vehicles from multiple depots are higher.

Zhong and Aghezzaf (2011) proposed a new variant of the vehicle routing problem called the "single vehicle cyclic inventory routing problem." The problem is formulated as a mixed integer programming model with linear constraints and a non-linear objective function. In this model, repeated distribution of a product from a single depot to a selected subset of retailers having stable demands takes place. The objective of this model is to determine the subset of retailers and quantity to be delivered to each retailer, and to minimize the total distribution and inventory cost. The authors proposed an optimization approach combining DC programming and branch-and-bound with a steep descent hybrid algorithm. The proposed algorithm worked efficiently for small problem sizes.

Chapter 4

Mathematical Formulation

In this chapter, a mixed integer linear programming (MILP) formulation of the problem is proposed. First, the required notation, including parameters and decision variables, are defined. Next, the complete MILP formulation is presented.

4.1 Notation – Parameters

In this formulation, R represents the set of all retailers and S represents the set of all suppliers. Each retailer $r \in R$ has a demand for products (goods) provided by supplier $s \in S$. This demand is given by D_{rs}^{prod}, and is expressed in units of pallets. Under a VMI system, where it is assumed that each supplier has a sufficient availability of goods, it is unnecessary to define a parameter for each supplier's available inventory levels. In other words, it is assumed that each retailer's stated demand may be satisfied by the suppliers.

However, the same assumption is not made for pallets. The reason for this difference is that accurate counts of empty pallet inventory may not be available, possibly due to the absence of a tracking mechanism, damage or theft of pallets, or the re-use of pallets by the retailer for other purposes. Thus, each retailer is responsible for reporting its availability of empty pallets. As a result, it is possible that a supplier's demand for empty pallets exceeds the available supply.

Each supplier $s \in S$ has a demand for empty pallets, D_s^{pal}. This demand may be satisfied by transporting available empty pallets from each retailer $r \in R$. The corresponding parameter is represented by A_{rs}^{pal}. The scaling parameter α captures the fraction of a fully-loaded pallet that is consumed by an empty pallet. For example, $\alpha = 1/10$ if 10 empty pallets occupy the space of 1 loaded pallet.

To facilitate the appropriate assignment of vehicles to cross docks, each cross dock is represented by two nodes in the network. The inbound node may be visited only by loaded vehicles, while the outbound node may be visited only by empty vehicles. The sets of all inbound and outbound nodes are given by CD^{in} and CD^{out}, respectively. Thus, the set of all cross dock nodes, represented by CD, is such that $CD = \{CD^{\text{in}} \cup CD^{\text{out}}\}$.

The proposed formulation utilizes a discretized time approach, such that the planning horizon is partitioned into discrete time intervals of equal duration. The width of each time interval may be a function of the length of the planning horizon. For example, 15-minute intervals may be appropriate for planning over an 8-hour horizon, while multi-day horizons may be more appropriately modeled with longer intervals. Let T represent the set of discrete time intervals comprising the planning horizon, where the interval $t_0 = \min\{T\}$ represents the start of the planning horizon. All vehicles are assumed to be at their initial locations and ready for service at time t_0. Although not considered in this formulation, it is possible to customize the model to account for a situation where each vehicle is ready for service at a unique time by changing this parameter to $t_{0,v}$, where v represents a particular vehicle.

Each supplier, retailer, and cross dock has an allowable time window in which service may begin. The time window for node j is represented by the set T_j. Because T_j is a discrete set, it is possible to include multiple disjoint time windows. This is a primary motivation for employing a discrete time formulation for this problem, as demonstrated in the constraints below.

Because the proposed model allows vehicles to travel multiple routes (or tours), a mechanism for tracking the number of routes assigned to each vehicle is required. This is accomplished by designating a unique identification (ID) number to each vehicle, such that this ID indicates the physical vehicle itself, as well as the route number. The identification scheme is managed as follows. First, let set V' represent each physical vehicle in the heterogeneous fleet, such that the cardinality of V' indicates the total number of vehicles in the fleet. Throughout this manuscript, elements of set V' may be referenced as "original" vehicles. For each vehicle $v \in V'$, a corresponding set, denoted as V''_v contains unique vehicle ID numbers for each additional route that may be taken by vehicle v. Vehicle IDs in set V''_v may be referenced as vehicle "copies." Thus, vehicle v's first route will be attributed to vehicle ID v, and v's second route will be attributed to the vehicle ID contained in the first element of V''_v. The cardinality of V''_v represents the number of additional routes that may be taken by vehicle $v \in V'$. Finally, the set V is defined to be the set of all unique vehicle IDs, such that $V = \{V' \cup V''_v\}$ for all $v \in V'$. Vehicle $v \in V$ has a capacity of C_v loaded pallets.

For an example of these vehicle sets, suppose a fleet contains two vehicles, such that $V' = \{1, 2\}$. Further, suppose vehicle 1 may take three additional tours, and vehicle 2 may take two additional tours. Thus, $V''_1 = \{3, 4, 5\}$ and $V''_2 = \{6, 7\}$.

The time required for vehicle v to travel from node i to node j is given by τ_{vij}, and is expressed in units of discrete time intervals. Any service duration required at node i (such as loading or unloading times) should be included in the value of τ_{vij}.

4.2 Notation – Network Description

To facilitate a complete description of the underlying network, additional notation characterizing the network nodes is beneficial. First, let Δ^0_v represent the initial location (at time t_0) of each vehicle $v \in V'$. Recall that V' is the set of all "original" vehicles. Because the "copies" of each vehicle have yet to be assigned, their initial locations are unknown *a priori*. This node may represent any location in space, and is not associated with any other node in the network.

Each vehicle is required to terminate each route to which it is assigned at a node in the set Δ^1. When vehicle $v \in V$ reaches a node in Δ^1, the vehicle receives a new identification number (from the set V''_v), and the vehicle is ready to start a new route. For example, vehicle $v \in V'$ starts at its initial location, Δ^0_v. It may visit several nodes in its first tour before terminating that tour at some node in the set Δ^1. The vehicle will then be assigned to the first element of the set V_v'', as its second tour will be attributed to the first "copy" of the vehicle. This first copy will begin its route at the node in Δ^1 where the "original" vehicle terminated, and it will terminate this route at any node in Δ^1. Upon returning to a node in Δ^1, the vehicle will be assigned to the next unique number in V_v''. This process repeats until all vehicle copies in V_v'' have been assigned.

By assumption (construction), each element $j \in \Delta^1$ should correspond to a cross dock location. Thus, three nodes are associated with each cross dock – a particular node $j \in CD^{\text{in}}$, a particular node $j \in CD^{\text{out}}$, and a particular node $j \in \Delta^1$.

Each vehicle route is defined to be a sequence of arcs in the network. For example, if a vehicle traverses arc (i, j) it departs from node i and travels directly to node j. Due to the specific nature of this problem, not all nodes in the network are directly connected. Therefore, two additional sets of nodes are defined to simplify the characterization of valid network arcs. The first such set, Δ_v^+, represents all nodes in the network to which vehicle v may travel. To afford maximum flexibility in the model, we define $\Delta_v^+ \subseteq \{\Delta^1 \cup CD \cup R \cup S\}$ for all $v \in V$. Note that $\Delta_v^0 \notin \Delta_v^+$ because vehicles may not travel to (i.e., return to) their initial location. The second set, Δ_{vj}^-, represents all nodes in the network from which vehicle v may travel to node $j \in \Delta_v^+$. The nodes contained in Δ_{vj}^- depend upon the classification of vehicle v (e.g., whether it is an "original" or "copy" vehicle) as well as the type of node j (e.g., whether it is a retailer or supplier node). Table 4.1 contains the definitions of Δ_{vj}^- for all possible v and j.

	Vehicle Classification	
Destination Node	$v \in V'$	$v \in V''$
$j \in \Delta^1$	$\Delta_{vj}^- \subseteq \{\Delta_v^0 \cup R \cup S \cup CD^{\text{in}}\}$	$\Delta_{vj}^- \subseteq \{j \cup R \cup S \cup CD^{\text{in}}\}$
$j \in R$	$\Delta_{vj}^- \subseteq \{\Delta_v^0 \cup R \cup CD^{\text{out}}\}$	$\Delta_{vj}^- \subseteq \{\Delta^1 \cup R \cup CD^{\text{out}}\}$
$j \in S$	$\Delta_{vj}^- \subseteq \{\Delta_v^0 \cup R \cup S\}$	$\Delta_{vj}^- \subseteq \{\Delta^1 \cup R \cup S\}$
$j \in CD^{\text{in}}$	$\Delta_{vj}^- \subseteq \{S\}$	$\Delta_{vj}^- \subseteq \{S\}$
$j \in CD^{\text{out}}$	$\Delta_{vj}^- = \Delta_v^0$	$\Delta_{vj}^- \subseteq \{\Delta^1\}$

Table 4.1: Possible definitions for Δ_{vj}^-

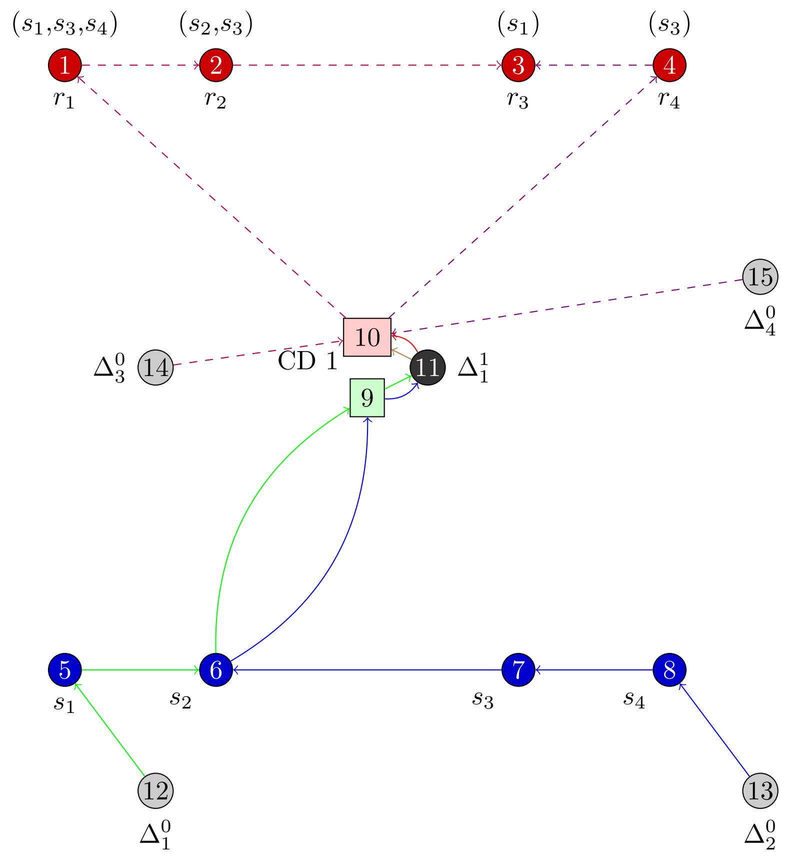

Figure 4.1: Sample network

Figure 4.1 shows a sample network, which includes four retailers ($R = \{1, 2, 3, 4\}$), four suppliers ($S = \{5, 6, 7, 8\}$), and one cross dock, decomposed into $CD^{\text{in}} = \{9\}$ and $CD^{\text{out}} = \{10\}$. The Δ^1 node, which is the node affiliated with the cross dock where vehicles are assigned new ID numbers, is represented by node $\Delta^1 = \{11\}$. The sample network has four original vehicles whose initial locations are defined by Δ_v^0 nodes, for all $v \in V$. The node numbers of the initial location are represented as $\Delta_1^0 = 12$, $\Delta_2^0 = 13$, $\Delta_3^0 = 14$, $\Delta_4^0 = 15$. Node 1 (retailer r_1) is to receive full pallets from suppliers s_1, s_2, and s_3. In Figure 4.1, this is denoted as (s_1, s_2, s_3) above node 1. Similarly, node 2 (r_2) receives full pallets from s_2 and s_3, node 3 (r_3) from s_1, and node 4 from s_3.

The routes taken by vehicles 1 and 2 are explained as follows. In Figure 4.1, vehicle 1 starts from its initial location (i.e., Δ_1^0 node), picks up full pallets from suppliers s_1, and

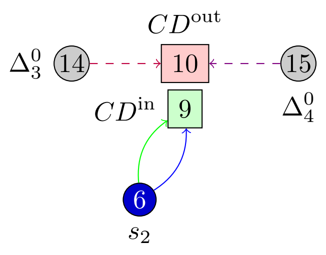

Figure 4.2: Activity in CD^{in}

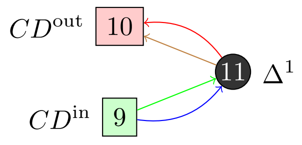

Figure 4.3: Activity in Δ^1

s_2, and reaches CD^{in}. Vehicle 2 starts from its initial location (i.e., Δ_2^0 node), picks up full pallets from suppliers s_4, s_3, and s_2 and reaches the CD^{in} node. Node 6 (s_2) is served by both vehicles 1 and 2 as part of a split pick up process.

Simultaneous arrival of vehicles at the cross dock is shown in Figure 4.2. Vehicles 1 and 2 travel from node 6 (s_2) to node 9 (CD^{in}). Meanwhile, vehicles 3 and 4 travel from their initial locations (i.e., Δ_3^0 and Δ_4^0, respectively) to node 10 (CD^{out}). At the cross dock, goods from incoming vehicles 1 and 2 are unloaded, consolidated, and loaded to the outbound vehicles 3 and 4. Since no inventory storage is allowed at the cross dock, consolidated goods are directly moved to the outbound trucks. For this event to happen, all vehicles should reach the cross dock at the same time interval, t_j^{max}, where $j \in CD$.

In Figure 4.3, vehicles 1 and 2 leave node 9 (CD^{in}) after unloading the full pallets, and travel to node 11 (Δ^1) to get assigned new vehicle numbers. The vehicles that leave the Δ^1 node are the copies of vehicles that enters the Δ^1 node. The vehicles coming out from the Δ^1 node will start the new cycle in similar manner to the assignments of vehicle 3 and 4.

The route taken by vehicle 3 is shown in Figure 4.4. Vehicle 3 starts from node 14 (Δ_3^0), visits retailers r_1, r_2, and r_3 to deliver full pallets picked up from CD^{out}. In Figure 4.4, (s_1, s_2, s_3) mentioned above the node 1 (r_1) explains that vehicle 3 delivers full pallets picked

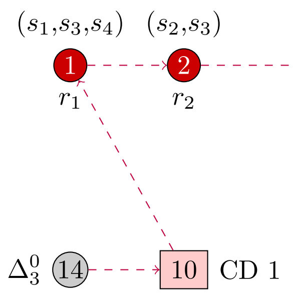

Figure 4.4: Route taken by vehicle 3

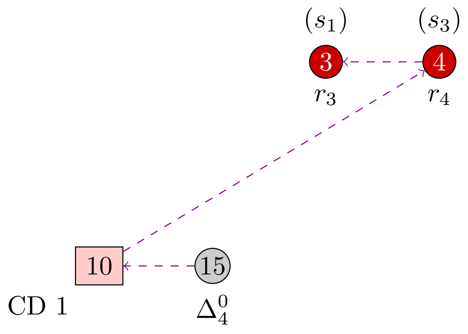

Figure 4.5: Route taken by vehicle 4

up from nodes s_1, s_2 and s_3 to r_1, as a result of consolidation process. The route taken by vehicle 4 is shown in Figure 4.5. Vehicle 4 starts from node 15 (Δ_4^0), visits retailers r_4 and r_3, to deliver full pallets picked up from CD^{out}. From Figures 4.4 and 4.5, it is observed that demand at node 3 (r_3), is fulfilled by both vehicles 3 and 4. This is an example of a split delivery operation. During the time of delivery of full pallets to retailers, empty pallets are picked up from the retailers. Then vehicles 3 and 4 follow the similar pattern of route of vehicles 1 and 2, to visit the supplier nodes to pick up full pallets and deliver empty pallets.

4.3 Notation – Decision Variables

Several types of decision variables are required to determine the route taken by each vehicle, the quantity of loaded pallets delivered from suppliers to retailers, the quantity of pallets transferred at the cross docks, and the quantity of empty pallets delivered from retailers to suppliers. Additionally, there are decision variables that capture the quantity of unmet demand requested by retailers and the number of empty pallets that cannot be delivered to suppliers. The decision variables are listed below.

x_{vij}^t Binary decision variable, such that $x_{vij}^t = 1$ if vehicle $v \in V$ travels from node i to node j and begins service at node j at time interval $t \in T$. Otherwise, $x_{vij}^t = 0$.

a_{vsr}^t Quantity of loaded pallets picked up from supplier $s \in S$ by vehicle $v \in V$ at time $t \in T_s$ that are bound for retailer $r \in R$.

b_{vsrj}^t Quantity of loaded pallets that are delivered by vehicle $v \in V$ to cross dock $j \in CD^{\text{in}}$ at time $t \in T_j$ from supplier $s \in S$ bound for retailer $r \in R$.

c_{vsrj}^t Quantity of loaded pallets that are picked up by vehicle $v \in V$ at cross dock $j \in CD^{\text{out}}$ at time $t \in T_j$ from supplier $s \in S$ bound for retailer $r \in R$.

q_{vsr}^t Quantity of loaded pallets delivered by vehicle $v \in V$ from supplier $s \in S$ to retailer $r \in R$.

e_{vsr}^t Quantity of empty pallets picked up by vehicle $v \in V$ at retailer $r \in R$ at time $t \in T_r$ bound for supplier $s \in S$.

f_{vs}^t Quantity of empty pallets delivered by vehicle $v \in V$ to supplier $s \in S$ at time $t \in T_s$.

$Z_{rs}^{\mathbf{prod}}$ Quantity of unmet demand by retailer $r \in R$ for products not delivered from supplier $s \in S$. Z_{rs}^{prod} is expressed in units of pallets.

$Z_s^{\mathbf{pal}}$ Number of empty pallets requested by supplier $s \in S$ that cannot be delivered.

4.4 Mathematical Model

The MILP formulation of this problem may be divided into five classes of constraints. These include network constraints describing valid routes for each vehicle, supplier constraints addressing the quantity of goods picked up from each supplier, retailer constraints ensuring appropriate deliver of goods to retailers, backhaul constraints relating empty pallets picked up from retailers to those delivered to suppliers, and cross dock constraints governing the coordination of vehicles at the cross dock locations. In a slight departure from convention, the objective function will be described after an explanation of the necessary constraints.

4.4.1 Network Constraints

The following network constraints ensure that each vehicle is assigned to valid routes, starting from the vehicle's initial location and ending with the last "copy" of each vehicle terminating at a node in Δ^1. These constraints are analogous to those found in traditional vehicle routing problems, and are adapted from the formulation of Murray and Karwan (2010), who studied the problem of unmanned aircraft routing.

The first two network constraints are concerned only with the initial route taken by each vehicle. In other words, only the "original" instance of each vehicle is considered, not "copies."

$$\sum_{j \in \{\Delta^+ \setminus CD^{\text{in}}\}} \sum_{t \in T_j} x^t_{v,\Delta^0_v,j} = 1 \qquad \forall\ v \in V' \tag{4.1}$$

$$\sum_{j \in \{\Delta^+ : \Delta^0_v \in \Delta^-_{vj}\}} \sum_{\substack{t \in T_j \\ t < t_0 + \tau_{v,\Delta^0_v,j}}} x^t_{v,\Delta^0_v,j} = 0 \qquad \forall\ v \in V' \tag{4.2}$$

Constraint (4.1) states that each vehicle $v \in V'$ must start at its initial location, Δ^0_v, and may travel to CD^{out}, a supplier in S, a retailer in R, or a final location in set Δ^1. Note that a vehicle may not travel directly from its initial location to a *loading* bay of a cross dock (CD^{in}) because the vehicle would arrive empty (nothing to unload). Also, if a vehicle is dispatched from its initial location to a retailer in R, the vehicle would arrive at the retailer with an empty truck. This would allow for the event that the vehicle is dispatched for the sole purpose of picking up empty pallets from one or more retailers (and then to deliver these empty pallets to one or more suppliers). If a vehicle is not assigned to any retailer or supplier nodes, it may be routed directly to a node in Δ^1.

Constraint (4.2) ensures that the first task assigned to each vehicle $v \in V'$ is at a feasible time. This takes into account both the allowable time windows for each node and the travel time required for a vehicle to travel from its initial location to a destination node.

The next constraint states that each vehicle in $V' \cup V''$ must end at a node in the set Δ^1:

$$\sum_{j \in \Delta^1} \sum_{i \in \Delta^-_{vj}} \sum_{t \in T_j} x^t_{vij} = 1 \qquad \forall\, v \in V \tag{4.3}$$

Note that Δ^-_{vj} contains the sets CD^{in}, R, and S. It also contains the vehicle's initial location, Δ^0_v, if $v \in V'$ or the nodes in Δ^1 if $v \in V''$. However, Δ^-_{vj} does not contain CD^{out} because we do not want to allow a fully-loaded vehicle to travel to a "final" location. Also, since the set S is included, it is possible for a vehicle to have visited suppliers without picking up any goods. This might occur if a vehicle is assigned only to deliver empty pallets to one or more suppliers.

Allowing vehicles to travel multiple routes necessitates specialized constraints to maintain the proper relationship among all routes. In these constraints, the "parent" vehicle is considered to be the instance of a vehicle that arrives at some node in Δ^1, while the "child" vehicle is the copy of that vehicle as it departs from the node in which the parent vehicle terminated. These constraints differ slightly between the case where the parent vehicle is an "original" vehicle and the case where the parent vehicle is a "copy." The constraints for managing the multiple use of vehicles are as follows:

$$\sum_{i\in\Delta^-_{v'j}}\sum_{t\in T_j} x^t_{v'ij} = \sum_{k\in\{\Delta^+_{v''}\setminus CD^{\text{in}}\}}\sum_{t\in T_k} x^t_{v''jk} \qquad \forall\ v'\in V', v''=V''_{v'}(1), j\in\Delta^1 \tag{4.4}$$

$$\sum_{j\in\Delta^1}\sum_{i\in\Delta^-_{v'j}}\sum_{t\in T_j} t x^t_{v'ij} \le \sum_{j\in\Delta^1}\sum_{k\in\{\Delta^+_{v''}:j\in\Delta^-_{v''k}\}}\sum_{t\in T_k}(t-\tau_{v''jk})\, x^t_{v''jk} \qquad \forall\ v'\in V', v''=V''_{v'}(1) \tag{4.5}$$

$$\sum_{i\in\Delta^-_{v'j}}\sum_{t\in T_j} x^t_{v'ij} = \sum_{k\in\{\Delta^+_{v''}\setminus CD^{\text{in}}\}}\sum_{t\in T_k} x^t_{v''jk}$$
$$\forall\ v\in V', m\in[1,|V''_v|-1], v'=V''_v(m), v''=V''_v(m+1), j\in\Delta^1 \tag{4.6}$$

$$\sum_{j\in\Delta^1}\sum_{i\in\Delta^-_{v'j}}\sum_{t\in T_j} t x^t_{v'ij} \le \sum_{j\in\Delta^1}\sum_{k\in\{\Delta^+_{v''}:j\in\Delta^-_{v''k}\}}\sum_{t\in T_k}(t-\tau_{v''jk})\, x^t_{v''jk}$$
$$\forall\ v\in V', m\in[1,|V''_v|-1], v'=V''_v(m), v''=V''_v(m+1) \tag{4.7}$$

Constraints (4.4) and (4.5) address the case where a vehicle terminates its first route and the first "copy" of the vehicle begins its route, such that $V''_{v'}(1)$ indicates the first element in the set $V''_{v'}$. Constraint (4.4) states that the first copy of vehicle v' must start at the same Δ^1 node where the corresponding original vehicle previously ended. Thus, this constraint ties the final location of a vehicle's first route with the initial location of its second route. Note that $k\in\{\Delta^+_{v''}\setminus CD^{\text{in}}\}$ is equivalent to $k\in\{\Delta^+_{v''}: j\in\Delta^-_{v''k}\}$. Constraint (4.5) prohibits the vehicle's second route from commencing before the termination of its first route. By itself, this constraint does not ensure that the "child" vehicle departs from the same location to which the "parent" vehicle terminated its tour. However, this requirement is handled by constraint (4.4).

Constraints (4.6) and (4.7) are the corresponding analogues for the subsequent case of both parent and child vehicles being copies. In these constraints, parameter m is employed to identify the proper element of the set V''_v.

An additional constraint is required to ensure that travel time is considered when creating vehicle routes, as described in constraint (4.8) below:

$$\sum_{i \in \Delta_{vj}^{-}} \sum_{t \in T_j} t x_{vij}^{t} \leq \sum_{k \in \{\Delta_v^{+} : j \in \Delta_{vk}^{-}\}} \sum_{t \in T_k} (t - \tau_{vjk})\, x_{vjk}^{t} \qquad \forall\, v \in V, j \in \{\Delta_v^{+} \setminus \Delta^{1}\} \tag{4.8}$$

If vehicle v travels arcs (i, j) and (j, k), service at node k cannot begin until the vehicle has traveled from j to k. Note that node j ignores nodes in the set Δ^1 because the same vehicle ID cannot be used to travel both to and from a final node. Travel time restrictions for these nodes are addressed in constraint (4.5).

Finally, the following three constraints complete the requirements for valid vehicle routes:

$$\sum_{j \in \{\Delta_v^{+} \setminus \Delta^{1} : t \in T_j\}} \sum_{i \in \Delta_{vj}^{-}} x_{vij}^{t} \leq 1 \qquad \forall\, v \in V, t \in T \tag{4.9}$$

$$\sum_{i \in \Delta_{vj}^{-}} \sum_{t \in T_j} x_{vij}^{t} = \sum_{k \in \{\Delta_v^{+} : j \in \Delta_{vk}^{-}\}} \sum_{t \in T_k} x_{vjk}^{t} \qquad \forall\, v \in V, j \in \{\Delta_v^{+} \setminus \Delta^{1}\} \tag{4.10}$$

$$\sum_{i \in \Delta_{vj}^{-}} \sum_{t \in T_j} x_{vij}^{t} \leq 1 \qquad \forall\, v \in V, j \in \Delta_v^{+} \tag{4.11}$$

Constraint (4.9) prohibits multitasking by ensuring that, at any time, a vehicle may visit no more than one node in the network. Constraint (4.10) states that vehicle v must enter and leave each node an equal number of times (possibly zero). This constraint works because we assume that a vehicle cannot revisit a node *during a single tour*. Constraint (4.11) prohibits vehicles from re-visiting any node during the course of a tour. Note that a particular vehicle may re-visit a node during a later tour, where the vehicle number is incremented when the vehicle passes through some node in Δ^1.

4.4.2 Supplier Pickup Constraints

The first supplier-specific constraint, (4.12) below, states that a vehicle cannot pickup goods at a supplier at time t unless the vehicle visits the supplier at time t:

$$\alpha f_{vs}^{t} + \sum_{r \in R} a_{vsr}^{t} \leq M_{4.12} \sum_{i \in \Delta_{vs}^{-}} x_{vis}^{t} \qquad \forall\, v \in V, s \in S, t \in T_s \tag{4.12}$$

In this constraint, $M_{4.12}$ is a sufficiently-large number, and is given by $M_{4.12} = C_v$ for a given v.

The following constraint ensures that vehicle capacity limitations are observed when a vehicle loads goods from a supplier location:

$$\sum_{r \in R} a_{vsr}^{t} \leq C_v - \sum_{s' \in S} \sum_{r \in R} \sum_{\substack{t' \in T_r \\ t' < t}} \alpha e_{vs'r}^{t'} + \sum_{s' \in S} \sum_{\substack{t' \in T_{s'} \\ t' \leq t}} \alpha f_{vs'}^{t'} - \sum_{\substack{s' \in S \\ s' \neq s}} \sum_{r \in R} \sum_{\substack{t' \in T_s \\ t' < t}} a_{vs'r}^{t'}$$
$$\forall\, v \in V, s \in S, t \in T_s \tag{4.13}$$

This constraint captures the fact that vehicles may load goods from suppliers as part of a linehaul operation, and may have also previously loaded empty pallets from retailers via a backhaul operation. In other words, the quantity of full pallets loaded at a supplier must not exceed (the capacity of the vehicle) − (the number of empty pallets that were picked up from retailers) + (the number of empty pallets that have been delivered to suppliers) − (full pallets that have already been loaded at other suppliers).

4.4.3 Retailer Delivery Constraints

The following constraints describe the delivery of products to retailers:

$$\sum_{v\in V}\sum_{t\in T_r} q^t_{vsr} + Z^{\text{prod}}_{rs} = D^{\text{prod}}_{rs} \qquad \forall\ r \in R, s \in S \tag{4.14}$$

$$\sum_{s\in S}\left(q^t_{vsr} + \alpha e^t_{vsr}\right) \le M_{4.15} \sum_{i\in\{CD^{\text{out}}\ \cup\ R\}} x^t_{vir} \qquad \forall\ v \in V, r \in R, t \in T_r \tag{4.15}$$

In constraint (4.14), each retailer expects to receive its full quantity of demand within its allowable time window, although split deliveries are permitted. In the event that it is infeasible to satisfy the retailer's demand, perhaps due to vehicle capacity limitations or time window conflicts, the value of Z^{prod}_{rs} becomes non-zero. This decision variable allows the constraint to be satisfied, but with a heavy penalty in the objective function. It should be noted that while the proper application of vendor managed inventory should guarantee that sufficient supplier inventory exists to satisfy the retailer's stated demand, other factors in the logistics system may prevent the delivery of some units. Constraint (4.15) states that delivery of goods to the retailer, and pickup of empty pallets from the retailer, may only occur when a vehicle visits the retailer. For this constraint, we may let $M_{4.15} = \lceil \min\left\{C_v, \sum_{s\in S} D^{\text{prod}}_{rs} + D^{\text{pal}}_{s}\right\}\rceil$ for a given r and v.

4.4.4 Backhaul Constraints

The incorporation of backhaul activities, where vehicles may pickup and deliver items at the same location, is a unique feature of this cross dock problem. The first three of four necessary constraints are as follows:

$$\sum_{v \in V} \sum_{t \in T_r} e^t_{vsr} \leq A^{\text{pal}}_{rs} \qquad \forall\ r \in R, s \in S \tag{4.16}$$

$$\sum_{s \in S} \sum_{t \in T_s} f^t_{vs} = \sum_{r \in R} \sum_{s \in S} \sum_{t \in T_r} e^t_{vsr} \qquad \forall\ v \in V \tag{4.17}$$

$$\sum_{v \in V} \sum_{t \in T_s} f^t_{vs} + Z^{\text{pal}}_s = D^{\text{pal}}_s \qquad \forall\ s \in S \tag{4.18}$$

Constraint (4.16) prevents a vehicle from picking up empty pallets in excess of what is available at each retailer, while constraint (4.17) requires that the quantity of empty pallets returned to a supplier by a particular vehicle must equal the quantity of empty pallets picked up from retailers by that vehicle. Finally, constraint (4.18) captures each supplier's demand for empty pallets, which must either be satisfied by actual pallets or covered by a non-zero value of Z^{pal}_s. Any positive value of Z^{pal}_s will be heavily penalized in the objective function.

The fourth backhaul constraint, which is analogous to constraint (4.13), prevents each vehicle from exceeding its capacity when loading empty pallets.

$$\sum_{s \in S} e^t_{vsr} \leq \alpha^{-1} C_v - \sum_{s \in S} \sum_{r' \in R} \left[\left(\sum_{j \in CD^{\text{out}}} \sum_{t' \in T_j} \alpha^{-1} c^{t'}_{vsr'j} \right) - \left(\sum_{\substack{t' \in T_{r'} \\ t' \leq t}} \alpha^{-1} q^{t'}_{vsr'} \right) + \left(\sum_{\substack{t' \in T_{r'} \\ t' < t}} e^{t'}_{vsr'} \right) \right]$$
$$\forall\ v \in V, r \in R, t \in T_r \tag{4.19}$$

Here, the quantity of empty pallets picked up by a vehicle must not exceed (the capacity of the vehicle) $-$ (the quantity of full pallets that were loaded on the vehicle at the cross dock) $+$ (the quantity of full pallets that have already been delivered) $-$ (the quantity of empty pallets that have already been picked up).

4.4.5 Cross Dock Constraints

The final set of constraints describe the coordination of vehicles, and the flow of goods, at the cross docks. These constraints are as follows:

$$b^t_{vsrj} \leq \sum_{t' \in T_s : t' \leq (t - \tau_{vsj})} a^{t'}_{vsr} \qquad \forall\ v \in V, r \in R, s \in S, j \in CD^{\text{in}}, t \in T_j \tag{4.20}$$

$$\sum_{s \in S} \sum_{r \in R} b^t_{vsrj} \leq M_{4.21} \sum_{s \in S} x^t_{vsj} \qquad \forall\ v \in V, j \in CD^{\text{in}}, t \in T_j \tag{4.21}$$

$$\sum_{s \in S} \sum_{r \in R} c^t_{vsrj} \leq M_{4.22} \sum_{i \in \Delta^-_{vj}} x^t_{vij} \qquad \forall\ v \in V, j \in CD^{\text{out}}, t \in T_j \tag{4.22}$$

$$\sum_{v \in V} c^t_{vsrj} = \sum_{v' \in V} b^t_{v'srj'} \qquad \forall\ s \in S, r \in R, (j, j') \in (CD^{\text{out}}, CD^{\text{in}}), t \in \{T_j \ \cap\ T_{j'}\} \tag{4.23}$$

$$\sum_{r \in R} \sum_{s \in S} \sum_{t \in T_r} q^t_{vsr} \leq C_v \qquad \forall\ v \in V \tag{4.24}$$

$$\sum_{t \in T_r} q^t_{vsr} = \sum_{j \in CD^{\text{out}}} \sum_{\substack{t' \in T_j \\ t' \leq \max\{T_r\}}} c^{t'}_{vsrj} \qquad \forall\ v \in V, s \in S, r \in R \tag{4.25}$$

Constraint (4.20) states that goods cannot be delivered to a cross dock unless they are first picked up from a supplier. Constraint (4.21) ensures that goods may only be delivered to the cross dock at time t if the vehicle visits the cross dock at time t. Similarly, constraint (4.22) prevents goods from being *loaded* at the cross dock at time t unless the vehicle visits the cross dock at time t. The values may be chosen such that $M_{4.21} = M_{4.22} = C_v$ for a given v.

One critical assumption of this model is that the cross dock does not hold any inventory. In other words, items must pass directly from the receiving docks to the shipping docks. To accomplish this, both the inbound and outbound trucks associated with a transfer of goods must be located at the cross dock simultaneously. This requirement is conveyed in constraint (4.23), which ensures that the appropriate quantity of goods are transferred at the cross dock at the proper time.

Once the vehicles are coordinated at the cross dock, constraint (4.24) prevents each vehicle from being loaded beyond capacity when receiving goods for delivery to retailers. Finally, constraint (4.25) ensures that the quantity of goods delivered to a retailer by a particular vehicle equals the quantity of goods picked up by that vehicle at the cross dock.

All decision variable definitions are described in (4.26) – (4.34), below:

$$x^t_{vij} \in \{0,1\} \qquad \forall\, v \in V, j \in \{\Delta^+_v\}, i \in \{\Delta^-_{vj}\}, t \in T_j \tag{4.26}$$

$$q^t_{vsr} \in \mathbb{Z}^+ \qquad \forall\, v \in V, s \in S, r \in R, t \in T_r \tag{4.27}$$

$$a^t_{vsr} \in \mathbb{Z}^+ \qquad \forall\, v \in V, s \in S, r \in R, t \in T_s \tag{4.28}$$

$$b^t_{vsrj} \in \mathbb{Z}^+ \qquad \forall\, v \in V, s \in S, r \in R, j \in CD, t \in T_j \tag{4.29}$$

$$c^t_{vsrj} \in \mathbb{Z}^+ \qquad \forall\, v \in V, s \in S, r \in R, j \in CD, t \in T_j \tag{4.30}$$

$$e^t_{vsr} \in \mathbb{Z}^+ \qquad \forall\, v \in V, s \in S, r \in R, t \in T_r \tag{4.31}$$

$$f^t_{vs} \in \mathbb{Z}^+ \qquad \forall\, v \in V, s \in S, t \in T_s \tag{4.32}$$

$$Z^{\text{prod}}_{rs} \in \mathbb{Z}^+ \qquad \forall\, r \in R, s \in S \tag{4.33}$$

$$Z^{\text{pal}}_{s} \in \mathbb{Z}^+ \qquad \forall\, s \in S \tag{4.34}$$

4.4.6 Objective Function

The objective of this problem is to determine minimum-cost vehicle routes and schedules, subject to the aforementioned constraints. The exact determination of the total cost may be application-specific. As such, the following definition of the objective function may require modification to support a particular organization's goals. One proposed calculation of total cost, TC, is as follows:

$$TC = \sum_{v\in V'} \sum_{k=1}^{|V''_v|} \sum_{i\in\Delta^1} \sum_{\substack{j\in\Delta^+_{V''_v(k)} \\ j\neq i}} \sum_{t\in T_j} kPx^t_{V''_v(k),i,j} + \sum_{v\in V} \sum_{j\in R} \sum_{i\in\Delta^-_{vj}} \sum_{t\in T_j} \left(t - \min\{T_j\}\right) x^t_{vij}$$
$$+ \sum_{r\in R} \sum_{s\in S} AZ^{\text{prod}}_{rs} + \sum_{s\in S} AZ^{\text{pal}}_{s} \tag{4.35}$$

The first term in this objective function reflects an increasing penalty for each extra tour made by a vehicle. In this term, k represents the index number of vehicle "copies," such that $V''_v(k)$ is the kth copy of vehicle v, and P represents a penalty for re-using vehicles. This term increases the vehicle utilization by minimizing the number of vehicle routes. The second term indicates our preference for serving each retailer at the earliest possible time within its allowable time window. This term is added to minimize the lead time.

The last two terms incorporate penalties for not meeting the demand of goods by retailers, and the demand of empty pallets by suppliers, respectively. In these terms, A represents the penalty for not meeting the demand. The value of A should be considered greater than the value of P. The main aim must be to transfer as many pallets as possible from suppliers to retailers. The cost of not having a product at a retailer when a consumer comes to buy a product will always be higher. Thus, the penalty for not delivering a pallet should always be higher. The P and A values may vary from one customer (i.e., a retail chain) to another.

Other objectives are possible, such as minimizing the number of "original" vehicles used, minimizing total travel distance, or minimizing the total cost of travel (where each arc on the network has an associated cost for each vehicle).

The MILP formulation for this cross dock-based distribution network may be stated as:

$$\begin{array}{ll} \text{Min} & TC \\ \text{s.t.} & \text{Constraints (4.1) – (4.34)} \end{array}$$

Chapter 5

Numerical Example

In this chapter, a sample problem is created to explain the mathematical model proposed in the previous chapter. This chapter is divided into two sections. Input data to the problem are explained in Section 5.1, while solutions are interpreted in Section 5.2.

5.1 Input Data

The sample problem includes four retailers, two suppliers, one cross dock, and four initial vehicles. Each vehicle may be reused one time (one "copy"). Each discrete time interval is assumed to be 15 minutes, and α is chosen as 0.1 (i.e., each full pallet consumes the same space as ten empty pallets).

The cross dock is decomposed into two nodes, CD^{in} and CD^{out}, such that CD^{in} is the location where full pallets are delivered and CD^{out} is the location where the consolidated full pallets are picked up.

Table 5.1 contains the unique node numbers assigned to all the nodes in the network. T_j represents the earliest and the latest time interval the vehicle can visit each node. Δ_v^0 is the notation for initial location of the vehicles. Δ_v^0 nodes do not have time intervals, because the vehicles cannot visit an initial location.

Figure 5.1 shows the location of nodes in the network, where $R = \{1, 2, 3, 4\}$, $S = \{5, 6\}$, $CD^{\text{in}} = \{7\}$, $CD^{\text{out}} = \{8\}$, $\Delta^1 = \{9\}$, $\Delta_1^0 = 10$, $\Delta_2^0 = 11$, $\Delta_3^0 = 12$, and $\Delta_4^0 = 13$.

Table 5.2 contains the A_{rs}^{pal} values. These values represent the demand for empty pallets that are available for pick up at retailer r to be delivered to supplier s.

Table 5.3 contains the D_{rs}^{prod} values. These values represent the demand for full pallets that are available for pick up at supplier s to be delivered to retailer r.

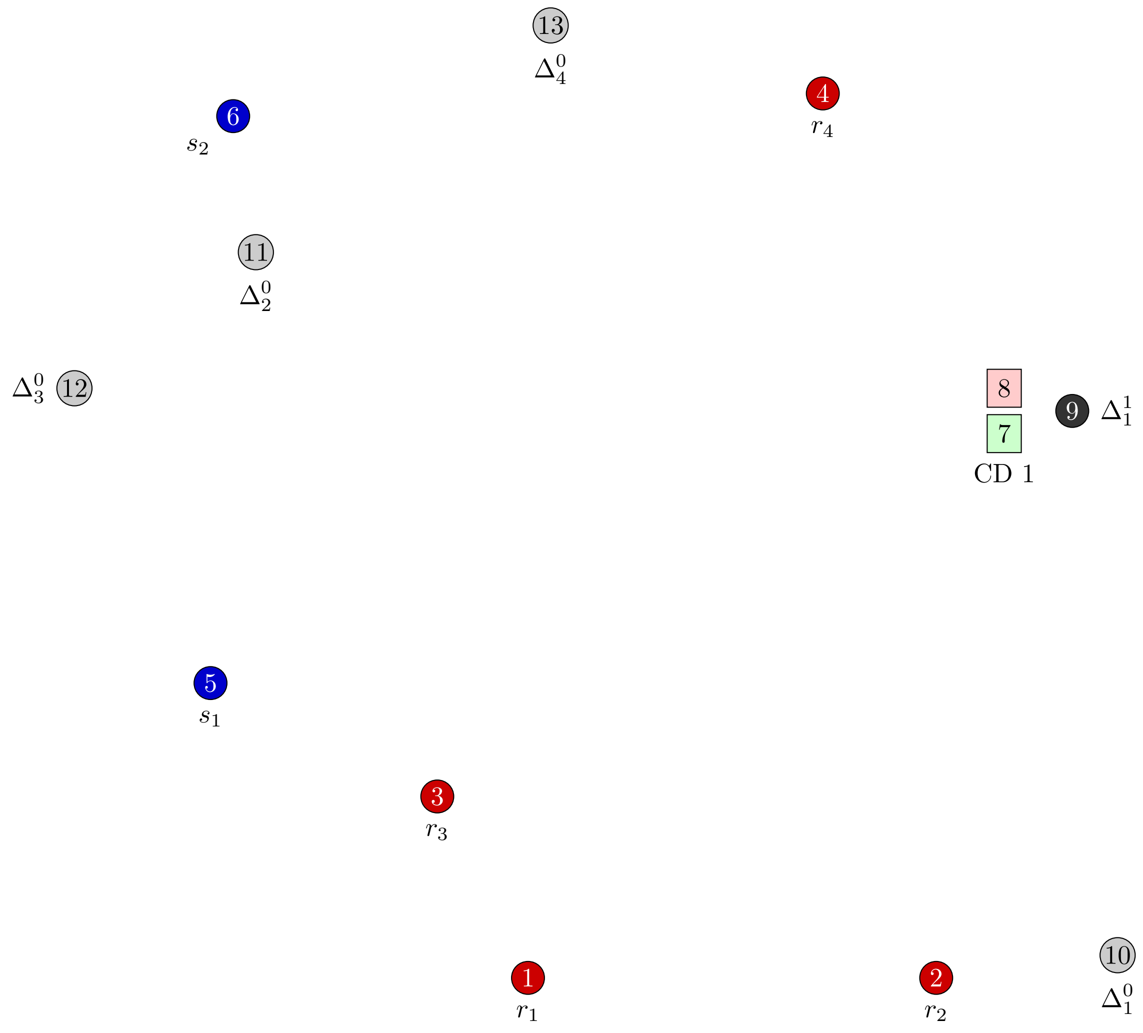

Figure 5.1: Nodes in the network

Node Name	Node #	T_j
Retailer # 1	1	[24,68]
Retailer # 2	2	[28,72]
Retailer # 3	3	[24,68]
Retailer # 4	4	[24,72]
Supplier # 1	5	[20,88]
Supplier # 2	6	[24,92]
CD^{in} # 1	7	[4,96]
CD^{out} # 1	8	[4,96]
Δ^1 # 1	9	[4,96]
Δ_1^0	10	
Δ_2^0	11	
Δ_3^0	12	
Δ_4^0	13	

Table 5.1: Location and time interval details

		$s \in S$	
		5	6
$r \in R$	1	32	38
	2	38	25
	3	2	35
	4	31	3

Table 5.2: A_{rs}^{pal} values

		$s \in S$	
		5	6
$r \in R$	1	3	2
	2	7	13
	3	6	12
	4	11	7

Table 5.3: D_{rs}^{prod} values

Vehicle #		C_v
V'	V''_v	
1	5	28
2	6	28
3	7	26
4	8	25

Table 5.4: C_v values

		j									
		1	2	3	4	5	6	7	8	9	10
i	1	-	3	2	7	4	7	-	-	5	-
	2	3	-	4	7	6	8	-	-	4	-
	3	2	4	-	6	2	5	-	-	5	-
	4	7	7	6	-	7	5	-	-	3	-
	5	-	-	-	-	-	5	6	-	6	-
	6	-	-	-	-	5	-	6	-	6	-
	7	-	-	-	-	-	-	-	-	1	-
	8	5	4	5	3	6	6	-	-	1	-
	9	5	4	5	3	6	6	1	1	0	-
	10	5	2	5	7	7	9	-	4	4	-

Table 5.5: τ_{1ij} values

Table 5.4 contains the C_v values, which reflect the capacity of each vehicle in terms of the number of full pallets. In this table, V' represents the original vehicles and V''_v represents the vehicle copy. In this problem $V' = \{1,2,3,4\}$ and $V''_v = \{5,6,7,8\}$.

Table 5.5 contains the τ_{1ij} values. It represents the time taken by the vehicle 1 to travel from start node i to destination node j. In the table, certain τ_{1ij} values are undefined (represented as "-") due to travel restrictions. Some of the travel restrictions include, a vehicle cannot travel between same nodes (i.e., $\tau_{1,1,1}$), with an exception for Δ^1 node. A vehicle cannot travel from a supplier node to a retailer node (i.e., $\tau_{1,5,1}$), a vehicle cannot travel from CD^{in} to R (i.e., $\tau_{1,7,3}$), a vehicle cannot travel from CD^{in} to S (i.e., $\tau_{1,7,5}$), a vehicle cannot travel from CD^{in} to CD^{out} (i.e., $\tau_{1,7,8}$) and a vehicle cannot travel from initial location (Δ_1^0) to CD^{in} (i.e., $\tau_{1,10,7}$).

		j									
		1	2	3	4	5	6	7	8	9	11
i	1	-	3	2	7	3	7	-	-	5	-
	2	3	-	4	6	6	8	-	-	4	-
	3	2	4	-	6	2	5	-	-	5	-
	4	7	6	6	-	6	5	-	-	3	-
	5	-	-	-	-	-	4	6	-	6	-
	6	-	-	-	-	4	-	6	-	6	-
	7	-	-	-	-	-	-	-	-	1	-
	8	5	4	5	3	6	6	-	-	1	-
	9	5	4	5	3	6	6	1	1	0	-
	11	6	7	4	5	3	1	-	6	6	-

Table 5.6: τ_{2ij} values

		j									
		1	2	3	4	5	6	7	8	9	12
i	1	-	4	2	8	4	8	-	-	6	-
	2	4	-	5	8	7	9	-	-	5	-
	3	2	5	-	7	3	6	-	-	6	-
	4	8	8	7	-	7	5	-	-	4	-
	5	-	-	-	-	-	5	7	-	7	-
	6	-	-	-	-	5	-	7	-	7	-
	7	-	-	-	-	-	-	-	-	1	-
	8	6	5	6	4	7	7	-	-	1	-
	9	6	5	6	4	7	7	1	1	0	-
	12	6	8	5	7	3	3	-	7	7	-

Table 5.7: τ_{3ij} values

		j									
		1	2	3	4	5	6	7	8	9	13
	1	-	3	2	7	4	7	-	-	6	-
	2	3	-	4	7	6	8	-	-	4	-
	3	2	4	-	6	2	5	-	-	5	-
	4	7	7	6	-	7	5	-	-	3	-
i	5	-	-	-	-	-	5	6	-	6	-
	6	-	-	-	-	5	-	6	-	6	-
	7	-	-	-	-	-	-	-	-	1	-
	8	6	4	5	3	6	6	-	-	1	-
	9	6	4	5	3	6	6	1	1	0	-
	13	7	8	6	3	6	3	-	5	5	-

Table 5.8: τ_{4ij} values

Tables 5.6 – 5.8 contain the τ_{vij} values for vehicles 2, 3 and 4, respectively. The only difference between these tables and Table 5.5 is in the initial location node, Δ_v^0, that occupies the last row and first column in the tables. The initial location for vehicles 2, 3 and 4 are 11, 12 and 13, respectively. The conditions for undefined values of τ_{vij} remain as previously described.

Since vehicles 5 – 8 are the copies of vehicles 1 – 4, respectively, the travel times for these vehicle copies are the same as the travel times for their corresponding "original" vehicles.

5.2 Solution

In this section, the solution generated by the CPLEX is explained. First, the route taken by each vehicle is explained. Sequentially in the following tables, the quantity of full pallets and empty pallets that are picked up and delivered by the vehicles in their route are explained.

Figure 5.2 and Table 5.9 show the route taken by each vehicle in the system. Additionally, Table 5.9 shows the sequence in which nodes are visited by each vehicle, as represented by x_{vij}^t variables that are assigned values of one. For example $x_{1,10,5}^{20}$ shows that vehicle 1 travels from node 10 and reaches node 5 at time 20. Next, $x_{1,5,6}^{25}$ shows vehicle 1 travels from node 5 and reaches node 6 at time 25. Each row in Table 5.9 shows the route taken by each vehicle. From the table we can infer that, in this problem, vehicles 1 – 5 are used for pick up and delivery operations, while vehicles 6, 7 and 8 are unused (their positions are shown at the Δ^1 node).

Vehicle #	Stop 1	Stop 2	Stop 3	Stop 4	Stop 5	Stop 6	Stop 7	Stop 8
1	$x^{20}_{1,10,5}$	$x^{25}_{1,5,6}$	$x^{31}_{1,6,7}$	$x^{34}_{1,7,9}$				
2	$x^{24}_{2,11,6}$	$x^{31}_{2,6,7}$	$x^{32}_{2,7,9}$					
3	$x^{31}_{3,12,8}$	$x^{35}_{3,8,4}$	$x^{42}_{3,4,3}$	$x^{45}_{3,3,5}$	$x^{50}_{3,5,6}$	$x^{58}_{3,6,7}$	$x^{84}_{3,7,9}$	
4	$x^{31}_{4,13,8}$	$x^{36}_{4,8,3}$	$x^{38}_{4,3,1}$	$x^{41}_{4,1,2}$	$x^{47}_{4,2,5}$	$x^{52}_{4,5,6}$	$x^{58}_{4,6,7}$	$x^{64}_{4,7,9}$
5	$x^{58}_{5,9,8}$	$x^{62}_{5,8,2}$	$x^{66}_{5,2,9}$					
6	$x^{56}_{6,9,9}$							
7	$x^{96}_{7,9,9}$							
8	$x^{96}_{8,9,9}$							

Table 5.9: Vehicle route sequence: x^t_{vij}

When a vehicle travels from one node to another, it performs either a pickup or delivery operation (or possibly both). From Table 5.9, $x^{20}_{1,10,5}$ shows that vehicle 1 travels from node 10 and reaches node 5 at time 20. From Table 5.10, we can see exactly what happens when the vehicle reaches node 5. For example $a^{20}_{1,5,1} = 3$ shows 3 full pallets of material are picked up by vehicle 1 at node 5 (a supplier) for node 1 (a retailer). Likewise, Table 5.10 shows the quantity of full pallets picked up by vehicle v for retailer r at supplier s at time t, shown as $a^t_{v,s,r}$.

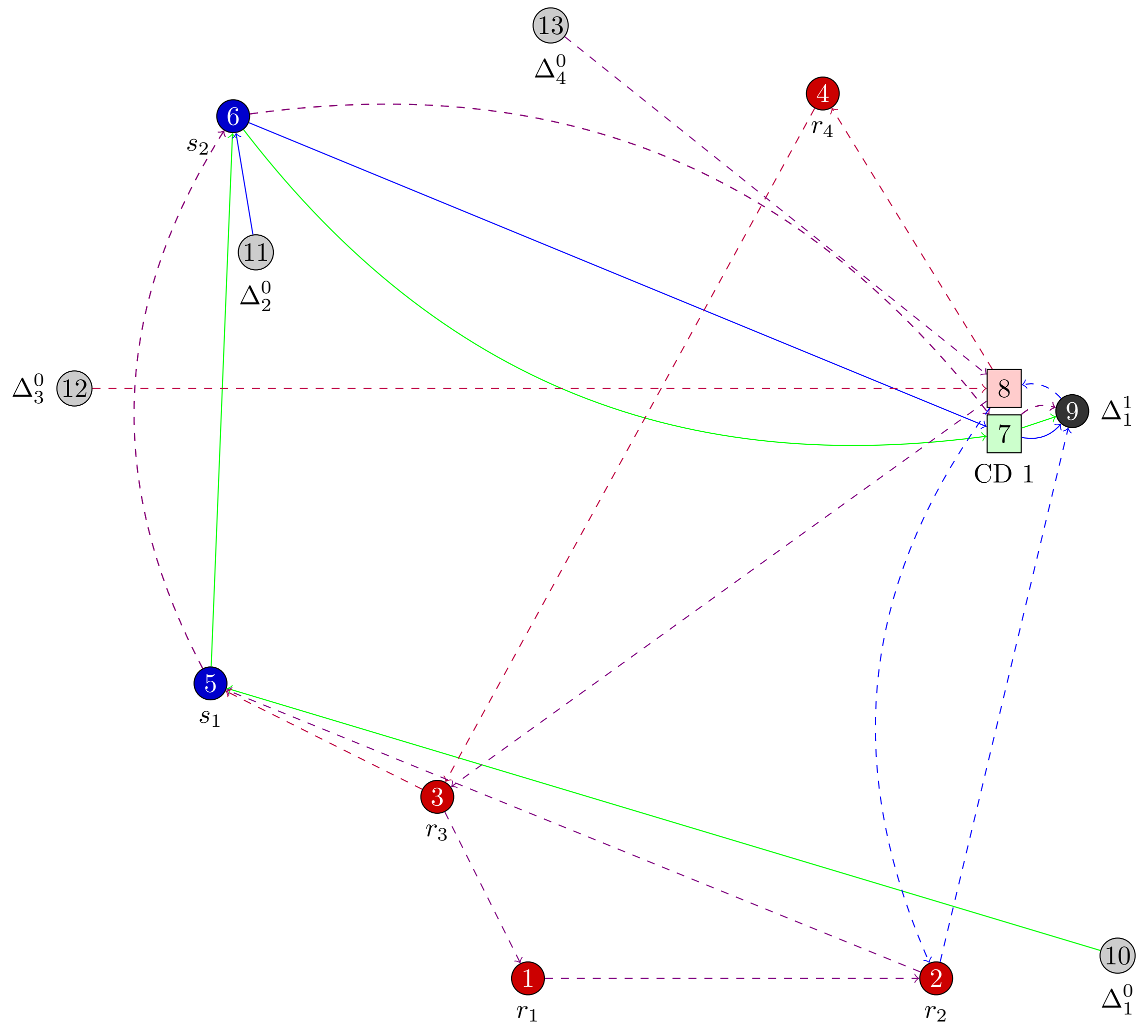

Figure 5.2: Routes taken by the vehicles in the network

a_{vsr}^{t}	# of Pallets
$a_{1,5,1}^{20}$	3
$a_{1,5,2}^{20}$	7
$a_{1,5,3}^{20}$	6
$a_{1,5,4}^{20}$	11
$a_{2,6,1}^{24}$	2
$a_{2,6,2}^{24}$	3
$a_{2,6,3}^{24}$	12
$a_{2,6,4}^{24}$	7
$a_{3,6,2}^{50}$	10

Table 5.10: Supplier full pallet pickup sequence

In Table 5.11, b_{vsrj}^{t} shows the quantity of full pallets delivered to cross dock j at time t by vehicle v that are picked up for retailer r at supplier s, and c_{vsrj}^{t} shows the quantity of full pallets picked up at the cross dock out j at time t by vehicle v for retailer r from supplier s. The events taking place at the cross dock can be explained as follows. If $b_{1,5,1,7}^{31} = 3$, then at time 31, vehicle 1 delivers 3 full pallets to node 7 (a cross dock), which is bound for node 1 (a retailer) from node 5 (a supplier). At the same time, event $c_{4,5,1,8}^{31} = 3$ takes place. This indicates that vehicle 4 picks up 3 full pallets at node 8 (a cross dock), which is bound for node 1 (a retailer) from node 5 (a supplier). From this table we can observe that goods brought by vehicle 1 are loaded to both vehicles 3 and 4, due to the consolidation process.

b^t_{vsrj}	# of Pallets	c^t_{vsrj}	# of Pallets
$b^{31}_{1,5,1,7}$	3	$c^{31}_{4,5,1,8}$	3
$b^{31}_{1,5,2,7}$	7	$c^{31}_{4,5,2,8}$	7
$b^{31}_{1,5,3,7}$	6	$c^{31}_{3,5,3,8}$	6
$b^{31}_{1,5,4,7}$	11	$c^{31}_{3,5,4,8}$	11
$b^{31}_{2,6,1,7}$	2	$c^{31}_{4,6,1,8}$	2
$b^{31}_{2,6,2,7}$	3	$c^{31}_{4,6,2,8}$	3
$b^{31}_{2,6,3,7}$	12	$c^{31}_{4,6,3,8}$	10
		$c^{31}_{3,6,3,8}$	2
$b^{31}_{2,6,4,7}$	7	$c^{31}_{3,6,4,8}$	7
$b^{58}_{3,6,2,7}$	10	$c^{58}_{5,6,2,8}$	10

Table 5.11: Cross dock in full pallet delivery and pick up sequence

q^t_{vsr}	# of Pallets
$q^{42}_{3,5,3}$	6
$q^{35}_{3,5,4}$	11
$q^{42}_{3,6,3}$	2
$q^{35}_{3,6,4}$	7
$q^{38}_{4,5,1}$	3
$q^{41}_{4,5,2}$	7
$q^{38}_{4,6,1}$	2
$q^{41}_{4,6,2}$	3
$q^{36}_{4,6,3}$	10
$q^{62}_{5,6,2}$	10

Table 5.12: Retailer full pallet delivery sequence

Table 5.12 shows the quantity of full pallets delivered to retailer r picked up from supplier s by vehicle v at time t, shown as q^t_{vsr}. Table 5.13 shows the quantity of empty pallets picked up at retailer r at time t to be delivered to supplier s by vehicle v, shown as e^t_{vsr}. The events taking place at the retailer node are explained as follows. From $q^{38}_{4,5,1} = 3$ and $e^{38}_{4,5,1} = 32$, we can infer that vehicle 4 reaches node 1 at time 38, delivers 3 full pallets and picks up 32 empty pallets. Events $q^{42}_{3,5,3}$ and $q^{36}_{4,6,3}$ are examples of split deliveries because demand at node 3 is satisfied by vehicles 3 and 4 at different time intervals.

e^t_{vsr}	# of Pallets
$e^{35}_{3,5,4}$	31
$e^{42}_{3,6,3}$	35
$e^{35}_{3,6,4}$	3
$e^{38}_{4,5,1}$	32
$e^{41}_{4,5,2}$	38
$e^{36}_{4,5,3}$	2
$e^{38}_{4,6,1}$	38
$e^{41}_{4,6,2}$	25

Table 5.13: Empty pallet pick up sequence

Table 5.14 shows the quantity of empty pallets delivered at supplier s by vehicle v at time t, shown as f^t_{vs}. For example $f^{50}_{3,6}$ shows that at time 50, 38 empty pallets are delivered by vehicle 3 to node 6.

f^t_{vs}	# of Pallets
$f^{45}_{3,5}$	31
$f^{50}_{3,6}$	38
$f^{47}_{4,5}$	72
$f^{52}_{4,6}$	63

Table 5.14: Empty pallet delivery sequence

Chapter 6

Heuristic Approach for Determining Initial Solutions

In this chapter a heuristic approach is proposed to obtain the initial feasible routes. During the validation stage, CPLEX gave no solution to the test problems, when stopped at the maximum run time of 30 minutes. The main reason for developing the heuristic is to obtain solutions at a faster rate when compared to CPLEX. The goal of the proposed heuristic is to determine an initial feasible, although likely sub-optimal, solution to the problem. Such a solution may be beneficial in the first step of a metaheuristic approach, such as tabu search.

The heuristic contains three functions:

1. `Pickup`: In this function, the vehicles are routed to the supplier to pick up full pallets of material.

2. `DeliveryAndPickup`: In this function, vehicles are routed to deliver full pallets to retailers and pick up empty pallets for the suppliers from the same retailer node. The vehicles are further routed to suppliers, to deliver empty pallets and pick up full pallets.

3. `VehicleCopy`: This function guides how the vehicle copies should behave.

This heuristic is restricted by three limitations. First, the number of "original" vehicles considered should always be an even number. Second, supplier and retailer nodes with the earliest time windows are visited first. For example, even if supplier s_1 with $D^{rs}_{prod} > 0$, is enroute to supplier s_2 (which has an earlier time window), supplier s_1 will not be visited. Finally, in a cycle, or wave, only half of the vehicle fleet is utilized. For example, if 10 vehicles are available, only 5 vehicles are used in a cycle for pick up and delivery operations while the other 5 vehicles will wait in their Δ^1 nodes. As per the cross dock constraint in the mathematical model, the transfer of pallets at a cross dock must take place from the inbound truck to outbound truck. To facilitate this feature, in this heuristic some vehicles are forced to wait at the cross dock. However in future research, the vehicles waiting at the cross dock can be utilized to make shorter routes, in addition to satisfying the cross dock constraints. This wave phenomenon occurs because all vehicles are assumed to be initially empty.

The heuristic begins by initializing several variables. First, the set of all "original" vehicles, V', is partitioned into two subsets, such that set PV represents the set of all vehicles that are available to be assigned to *pickup* full pallets from suppliers and set DV is the set of vehicles that are available to pickup goods from a cross dock and *deliver* these to retailers. These sets are initialized as follows:

$$PV = \left\{1, 2, \dots, \left\lfloor \frac{|V'|}{2} \right\rfloor \right\},$$
$$DV = V' \setminus PV.$$

Every cycle will end at the Δ^1 node and a new vehicle ID number is generated for each vehicle at this node. Set $DV1$ contains the copies of vehicles that are created at the end of function `Pickup`, and $DV2$ contains the copies of vehicle created at the end of function `DeliveryAndPickup`.

Next, the initial location and the starting time for each vehicle must be initialized:

$$myPrevNode_v = \Delta_v^0 \ \forall \ v \in V',$$
$$t_v = t_0 \ \forall \ v \in V'.$$

Third, let $D'^{\text{prod}}_{r,s}$ represent the unsatisfied demand of goods by retailer r from supplier s. This value will be updated as the heuristic progresses, and is initialized to equal the corresponding value of $D^{\text{prod}}_{r,s}$:

$$D'^{\text{prod}}_{r,s} = D^{\text{prod}}_{r,s} \; \forall \; r \in R, s \in S.$$

Fourth, let $A'^{\text{pal}}_{r,s}$ represent the unsatisfied demand of empty pallets by supplier s from retailer r. This value will be updated as the heuristic progresses, and is initialized to equal the corresponding value of $A^{\text{pal}}_{r,s}$:

$$A'^{\text{pal}}_{r,s} = A^{\text{pal}}_{r,s} \; \forall \; r \in R, s \in S.$$

Finally, to facilitate the coordinated arrival of vehicles at each cross dock, let $t^{\max}_{j}$ represent the latest time that an active vehicle is assigned to visit cross dock $j \in CD^{\text{in}}$. This value will be updated by the heuristic, and is initialized to equal the starting time interval of the planning horizon:

$$t^{\max}_{j} = t_0 \; \forall \; j \in CD^{\text{in}}.$$

The remainder of this chapter provides detailed descriptions of the three functions employed by the proposed heuristic.

6.1 The `Pickup` function

Step 1. In this step, each vehicle in set PV will be assigned to pick up full pallets from suppliers. Goods will be loaded onto each of these vehicles until either the vehicle contains no excess capacity, or there are no more goods to pick up. When vehicle v is assigned to visit a particular eligible supplier $j \in S'$, after having previously visited node $i \in \Delta^{-}_{v,j}$, the heuristic sets the value of $x^{t}_{v,i,j}$, where t represents the earliest feasible time that the vehicle can visit node j. Here, the set S' represents all suppliers that may be visited within their time windows and which hold goods for retailers whose time windows have yet to close. The process of assigning the vehicle to suppliers repeats until the vehicle has no further suppliers to visit, at which point the vehicle is routed to the nearest cross dock. Once all vehicles have been routed to a cross dock, the vehicles are routed to the appropriate node in set Δ^1, where the vehicle ends its route and is assigned a new vehicle number.

```
for all $v \in PV$ do
  $i = myPrevNode_v$; // Set the initial location for this vehicle.
  $LQ_v = 0$; // Loaded quantity of vehicle $v$ is zero.
  while $LQ_v < c_v$ do
    // Define the set of eligible suppliers:
      $S' = \left\{ s \in S : t_v + \tau_{v,i,s} \leq \max\{T_s\}, \sum_{\substack{r \in R \\ t_v \leq \max\{T_r\}}} D'^{\text{prod}}_{r,s} > 0 \right\}$;
    // Choose the eligible supplier that can be visited earliest.
    // Break ties by selecting the eligible supplier with the smallest ID:
      $j \leftarrow \min \left\{ s \in S' : \min\{T_s\} = \min_{k \in S'} \{\min\{T_k\}\} \right\}$;
    // Load goods bound for retailer $r$ from supplier $j$:
    for all $r \in R$ do
      if $(t_v \leq \max\{T_r\})$ then
        $LQ_v \leftarrow LQ_v + \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$; // Update loaded quantity on vehicle $v$.
        $Q^{\text{pick}}_{v,r,j} \leftarrow Q^{\text{pick}}_{v,r,j} + \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$; // Update quantity picked up.
        $D'^{\text{prod}}_{r,j} \leftarrow D'^{\text{prod}}_{r,j} - Q^{\text{pick}}_{v,r,j}$; // Update remaining availability.
      end if
    end for
    $t_v \leftarrow \max\{t_v + \tau_{v,i,j}, \min\{T_j\}\}$; // Find earliest feasible time to visit supplier $j$.
    Set $x^{t_v}_{v,i,j} = 1$; // Assign $v$ to visit supplier $j$ at a feasible time.
```

```
        $i \leftarrow j$; // Update the current location.
      end while
      // Find nearest cross dock $j \in CD^{\text{in}}$ such that $\tau_{v,i,j} = \min_{k \in CD^{\text{in}}} \{\tau_{v,i,k}\}$.
      $myCD_v \leftarrow j$; // Store the node number of the chosen CD for Step 2.
      $myPrevNode_v \leftarrow i$; // Store the previous node for Step 2.
      $t_j^{\max} \leftarrow \max\{t_j^{\max}, t_v + \tau_{v,i,j}\}$; // Update the latest time a vehicle is assigned to
      this CD.
    end for
```

Step 2. At this point, we have determined the earliest possible time that each "pickup" vehicle may be routed to a cross dock. Now, we must coordinate the arrival times of these vehicles at each cross dock. After visiting the cross dock, the vehicles are routed to a Δ^1 node to get assigned a new vehicle ID number. Each of the vehicles in set PV, after getting assigned a new number, is added to set DV1. The set DV1 contains the copies of vehicle, and is updated every cycle (when new vehicle copies are generated).

```
    $Q_{j,r,s}^{\text{in}} = 0 \ \forall \ j \in CD^{\text{in}}, r \in R, s \in S$; // Initialize the quantity of goods arriving at
    the cross dock.
    for all $v \in PV$ do
      $i \leftarrow myPrevNode_v$; // Last supplier visited.
      $j \leftarrow myCD_v$; // Selected cross dock.
      $t_v \leftarrow t_j^{\max}$; // Coordinated time for visiting this cross dock.
      Set $x_{v,i,j}^{t_v} = 1$; // Assign $v$ to visit cross dock at appropriate time.
      // Route $v$ to the appropriate $\Delta^1$ node, ending its tour:
          $t_v \leftarrow t_v + \tau_{v,j,\Delta_j^1}$; // Updated time.
          Set $x_{v,j,\Delta_j^1}^{t_v} = 1$; // Assign $v$ to final node.
      $v' \leftarrow$ next copy of vehicle $v$;
      $t_{v'} = t_v$; // Initialize the starting time of vehicle $v'$.
```

```
        $myPrevNode_{v'} = \Delta_j^1$; // Initialize starting node of $v'$.
        // Add the vehicle copy to the set DV1.
        Set $DV1 \leftarrow v'$; //Update $DV1$, to be used in Function Vehicle Copy
    **end for**
```

Step 3. In a cross dock, no storage of pallets are allowed, by assumption. Therefore, all pallets brought by the inbound trucks must be transferred to outbound trucks. To satisfy this condition, the total quantity of pallets brought by the inbound trucks should not exceed the total capacity of the outbound trucks at a cross dock. The vehicles are paired, such that if the first vehicle in set PV reaches CD_1^{in}, then the first vehicle in set DV should reach CD_1^{out}. To satisfy this condition, first the quantity of full pallets brought by the inbound trucks to each CD^{in} is calculated, and then the quantity of pallets that can be loaded to the outbound trucks is calculated. The excess pallets (i.e., the difference between the loaded quantity of vehicles and the capacity of their paired vehicle at a particular cross dock) that cannot be loaded to outbound trucks are added back to $D'^{\text{prod}}_{r,j}$. For example, if 25 pallets are brought by vehicle 1 and 28 by vehicle 2 to CD_1^{in}, and if the maximum capacity of both outbound truck is 26 pallets each, then the difference in capacity of 1 pallet (i.e., 26+26-28-25) is again added back to the respective $D'^{\text{prod}}_{r,j}$ value.

```
    // Initialize the quantity of pallets in CD
    $Q_j^{\text{CDout}} = 0 \ \forall \ j \in CD^{\text{out}}$; // Initialize the quantity of goods in the cross dock out.
    // Calculate the full pallets brought to $CD^{\text{in}}$ and quantity that can be loaded at $CD^{\text{out}}$
    **for all** $v \in PV$ **do**
        **for all** $v' \in DV$ **do**
            **if** (v and v' are the $n^{th}$ vehicle in the sequence) **then**
                $j \leftarrow myCD_v$; // Selected cross dock.
                $Q_j^{\text{CDin}} = Q_j^{\text{CDin}} + LQ_v$; // Update quantity in $CD^{\text{in}}$
            $Q_j^{\text{CDout}} = Q_j^{\text{CDout}} + C_{v'}$; // Update available capacity at $CD^{\text{out}}$
        **end if**
    **end for**
**end for**
// Transfer the excess capacity to $D'^{\text{prod}}_{r,s}$.
**for all** $v \in PV$ **do**
    **for all** $j \in CD^{\text{in}}$ **do**
        **while** $((Q_j^{\text{CDin}} > Q_j^{\text{CDout}}))$ **do**
            // Choose the eligible supplier to update the $D'^{\text{prod}}_{b,a}$ value
            // Break ties by selecting the eligible supplier with the largest ID:
                $a \leftarrow \max\left\{s \in S : \min\{T_s\} = \max_{k \in S}\{\min\{T_k\}\}\right\}$;
            // Choose the eligible retailer to update the $D'^{\text{prod}}_{b,a}$ value
            // Break ties by selecting the eligible retailer with the largest ID:
                $b \leftarrow \max\left\{r \in R : \min\{T_r\} = \max_{n \in R}\{\min\{T_n\}\}\right\}$;
            **if** $((Q_j^{\text{CDin}} - Q_j^{\text{CDout}}) \geq Q^{\text{pick}}_{v,b,a})$ **then**
                // If full pallets picked up at a supplier for a retailer is less than the quantity
                to be transferred, then
                $D'^{\text{prod}}_{b,a} \leftarrow D'^{\text{prod}}_{b,a} + Q^{\text{pick}}_{v,b,a}$; // Update the $D'^{\text{prod}}_{b,a}$ value
                $Q_j^{\text{CDin}} \leftarrow Q_j^{\text{CDin}}$ - $Q^{\text{pick}}_{v,b,a}$ ; // Update the full pallet capacity at $CD^{\text{in}}$
                $Q^{\text{pick}}_{v,b,a}$ =0; // Update the full pallet quantity picked up.
            **else**
                // If full pallets picked up at a supplier for a retailer is greater than the
                quantity to be transferred, then
                $D'^{\text{prod}}_{b,a} \leftarrow D'^{\text{prod}}_{b,a}$ + ($Q_j^{\text{CDin}}$ - $Q_j^{\text{CDout}}$); // Update the $D'^{\text{prod}}_{b,a}$ value
                $Q_j^{\text{CDin}} \leftarrow Q_j^{\text{CDin}}$ - ($Q_j^{\text{CDin}}$ - $Q_j^{\text{CDout}}$) ; // Update the full pallet capacity at
                $CD^{\text{in}}$
            $Q^{\text{pick}}_{v,b,a} \leftarrow Q^{\text{pick}}_{v,b,a}$ - ($Q^{\text{CDin}}_{j}$ - $Q^{\text{CDout}}_{j}$) ; // Update the full pallet quantity picked up.
          end if
        end while
      end for
    end for
    //Update the quantity of goods arriving at CD
    $Q^{\text{in}}_{j,r,s} = 0 \; \forall \; j \in CD^{\text{in}}, r \in R, s \in S$; // Initialize the quantity of goods arriving at the cross dock.
    for all $v \in PV$ do
      $j \leftarrow myCD_v$; // Selected cross dock.
      for all $r \in R$ do
        for all $s \in S$ do
          $Q^{\text{in}}_{j,r,s} \leftarrow Q^{\text{in}}_{j,r,s} + Q^{\text{pick}}_{v,r,s}$; // Update quantity of goods arriving at cross dock.
        end for
      end for
    end for
```

6.2 The `DeliveryAndPickup` function

Step 1. In this step, goods are first transferred from the unloading to loading bay of the cross dock. Then, each vehicle in set DV will be assigned to pick up full pallets from the cross dock. For example, if there are two cross docks and, from the previous function, if two vehicles reach CD^{in}_{1} and three vehicles reach CD^{in}_{2}, then in this function two vehicles should be routed to CD^{out}_{1} and three vehicles to CD^{out}_{2}. The vehicles should reach the corresponding cross dock at time $t^{\max}_{j}$, $j \in CD$. Goods are then loaded onto each of these vehicles until either the vehicle contains no excess capacity, or there are no more goods to pick up. The set R' represents all retailers that should be visited to deliver the full pallets. At the cross dock, goods are first loaded for the retailer with the earliest time window. After goods are loaded, the heuristic first sets the value of $x^t_{v,i,j}$ for vehicles with $LQ_v > 0$, where t represents the coordinated time for visiting the cross dock, and i is the previous node for the vehicle, and $j \in CD^{out}$. Next, for all vehicles such that $LQ_v = 0$, the heuristic sets the value of $x^t_{v,i,j}$ in Step 4. Here i and j refer to the Δ^1 node. Only vehicles which are loaded with full pallets must start the route. A variable, $Counter_v$, is introduced to differentiate the loaded and unloaded vehicles. $Counter_v$ is initialized as 1 if the vehicle is loaded, and as 0 if the vehicle is not loaded.

```
// Transfer the goods from $CD^{in}$ to $CD^{out}$
for all $k \in CD^{in}$ do
  for all $l \in CD^{out}$ do
    if (k and l are the $CD^{in}$ and $CD^{out}$ values of same cross dock) then
      for all $r \in R$ do
        for all $s \in S$ do
          $Q^{\text{out}}_{l,r,s} \leftarrow Q^{\text{in}}_{k,r,s}$ ; // Transfer goods from $CD^{in}$ to $CD^{out}$.
        end for
      end for
      // Find the appropriate CD for vehicles.
      for all $v' \in PV$ do
        for all $v \in DV$ do
          $i \leftarrow myPrevNode_v$; // Set the initial location for this vehicle.
          $LQ_v = 0$; // Loaded quantity of vehicle $v$ is zero.
          if (v and v' are the $n^{th}$ vehicle in the sequence) then
            if ($k == myCD^{in}_{v'}$) then
              $myCD^{out}_v \leftarrow l$; // Initialize the Cross dock.
              $j \leftarrow myCD^{out}_v$; // Initialize the Cross dock.
            $t_j^{\max} \leftarrow t_k^{\max}$; // Assign the max time value at $CD^{\text{out}}$
            $t_v \leftarrow t_j^{\max}$; // Coordinated time for visiting this cross dock
            // Load goods bound for retailer $r$ from CD:
            // Define the set of eligible retailers
                $R' = \left\{ r \in R : \sum_{s \in S} \sum_{l \in CD^{out}} Q_{l,r,s}^{\text{out}} > 0 \right\}$;
            for all $r \in R'$ do
                for all $s \in S$ do
                    while $LQ_v < c_v$ do
                        $Q_{v,r,s}^{\text{pick}} \leftarrow \min\{Q_{l,r,s}^{\text{out}}, C_v - LQ_v\}$ ; // Update quantity picked
                        up by vehicle
                        $Q_{l,r,s}^{\text{out}} \leftarrow Q_{l,r,s}^{\text{out}} - Q_{v,r,s}^{\text{pick}}$ ;// Update quantity at cross dock.
                        $LQ_v \leftarrow LQ_v + Q_{v,r,s}^{\text{pick}}$ ; // Update loaded quantity in vehicle
                    end while
                end for
            end for
            if $(LQ_v > 0)$ then
                Set $x_{v,i,j}^{t_v} = 1$; // Assign $v$ to visit cross dock at appropriate time.
                $myPrevNode_v \leftarrow j$; // Set the previous node for Step 2.
                $Counter_v = 1$; // Initialize the $Counter_v$ value.
            end if
            if $(LQ_v == 0)$ then
                $Counter_v = 0$; // Initialize the $Counter_v$ value.
            end if
        end if
    end if
  end for
end for
```

```
        end if
      end for
    end for
```

Step 2. In this step, each vehicle in set DV will be assigned to deliver full pallets and pick up empty pallets at retailers. Goods will be loaded onto each of these vehicles until either the vehicle contains no excess capacity, or there are no more empty pallets to pick up. When vehicle v is assigned to visit a particular eligible retailer $j \in R'$, after having previously visited node $i \in \Delta^-_{v,j}$, the heuristic sets the value of $x^t_{v,i,j}$, where t represents the earliest feasible time that the vehicle can visit node j. Here, the set R' represents all retailers for whom the full pallets are picked up. The process of assigning the vehicle to retailers repeats until all the full pallets picked up by the vehicles are delivered. Empty pallets are only picked up from the retailers who also receive full pallets. After all the full pallets are delivered, the vehicles are routed to the appropriate nodes in the set Δ^+_v.

```
    for all $v \in DV$ do
      if ($Counter_v > 0$) then
        // Choose the eligible retailer that can be visited earliest from set R'.
        // Break ties by selecting the eligible retailer with the smallest ID:
          $j \leftarrow \min\left\{ r \in R' : \min\{T_r\} = \min_{k \in R'}\{\min\{T_k\}\} \right\}$;
          $t'_v$=max$\{(t_v + \tau_{vij}),\ \min\{T_j\}\}$;
        if ($t'_v \leq \max\{T_j\}$) then
          // Deliver full pallets to retailers.
          for all $s \in S$ do
            if ($Q^{\text{pick}}_{v,j,s} > 0$) then
              $LQ_v \leftarrow LQ_v - Q^{\text{pick}}_{v,j,s}$; // Update loaded quantity in vehicle
              $Q^{\text{pick}}_{v,j,s} \leftarrow 0$; // Reset the full pallet quantity to zero
              // If current time is less than the maximum time interval at $S$
          if ($t'_v \leq \max\{T_s\}$) then
            // Pick up empty pallets at retailers.
            if ($A'^{\text{pal}}_{j,s} > 0$) then
              while $LQ_v < c_v$ do
                // Update loaded quantity on vehicle $v$:
                $LQ_v \leftarrow LQ_v + \min\{(A'^{\text{pal}}_{j,s} * \alpha), c_v - LQ_v\}$;
                // Update empty pallets picked up:
                $Q^{\text{empty}}_{v,j,s} \leftarrow \min\{(A'^{\text{pal}}_{j,s} * \alpha), c_v - LQ_v\}$;
                // Update remaining empty pallet quantity:
                $A'^{\text{pal}}_{j,s} \leftarrow A'^{\text{pal}}_{j,s} - ((\min\{(A'^{\text{pal}}_{j,s} * \alpha), c_v - LQ_v\})/\alpha)$;
              end while
            end if
          end if
          // Assign the $x^{t_v}_{v,i,j}$ values if vehicle visits retailer.
          if ($Q^{\text{pick}}_{v,j,s} == 0$) then
            $i \leftarrow myPrevNode_v$; // Set the previous node for Step 2.
            // Coordinated time for visiting this cross dock
            $t_v \leftarrow max\{(t_v + \tau_{vij}), min\{T_j\}\}$;
            Set $x^{t_v}_{v,i,j} = 1$; // Assign $v$ to visit retailer at appropriate time.
            $myPrevNode_v \leftarrow j$; // Set the previous node for Step 3.
          end if
        end if
      end for
    end if
  end if
end for
```

Step 3. In this step, two sets, S'' and S''' are first defined. Here set S'' represents all suppliers for whom the empty pallets are picked up at retailers, and set S''' represents the set of suppliers for whom no empty pallets are picked up at retailers. Each vehicle in set DV will be assigned to deliver empty pallets and pick up full pallets first at the supplier S''. If $LQ_v < C_v$, vehicles are then assigned to visit suppliers in set S''' for pick up. Here the goods will be loaded onto each of these vehicles until either the vehicle contains no excess capacity, or there are no more empty pallets to pick up. The process of assigning the vehicle to suppliers repeats until either the vehicle contains no excess capacity or has no further suppliers to visit. When vehicle v is assigned to visit a particular eligible supplier $j \in S$, where $S = \{S'' \cup S'''\}$, after having previously visited node $i \in \Delta^-_{v,j}$, the heuristic sets the value of $x^t_{v,i,j}$, where t represents the earliest feasible time that the vehicle can visit node j.

```
for all $v \in DV$ do
  if $(Counter_v > 0)$ then
    // Define the sets of eligible suppliers
      $S'' = \left\{ s \in S : \sum_{r \in R} \sum_{v \in DV} Q^{\text{empty}}_{v,r,s} > 0 \right\};$
      $S''' = \left\{ s \in S : \sum_{r \in R} \sum_{v \in DV} Q^{\text{empty}}_{v,r,s} = 0 \right\};$
    // Deliver empty pallets to suppliers in set $S''$ and pick up full pallets.
    // Choose the eligible supplier that can be visited earliest from set S”.
    // Break ties by selecting the eligible retailer with the smallest ID:
      $j \leftarrow \min \left\{ s \in S'' : \min\{T_s\} = \min_{k \in S''}\{\min\{T_k\}\} \right\};$
      $t'_v$=max$\{(t_v + \tau_{vij}), \min\{T_j\}\};$
    if $(t'_v \leq \max\{T_j\})$ then
      for all $r \in R$ do
        if $(Q^{\text{empty}}_{v,r,j} > 0)$ then
          $LQ_v \leftarrow LQ_v - Q^{\text{empty}}_{v,r,j}$; // Update loaded quantity in vehicle
      $Q^{\text{empty}}_{v,r,j} \leftarrow 0$; // Reset the empty pallet quantity to zero
      // Load goods bound for retailer $r$ from supplier $j$:
      **while** $(LQ_v < c_v)$ **do**
        **if** $(t'_v \leq \max\{T_r\})$ **then**
          // Update loaded quantity on vehicle $v$.
          $LQ_v \leftarrow LQ_v + \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$;
          // Update quantity picked up at each supplier.
          $Q^{\text{pick}}_{v,r,j} \leftarrow Q^{\text{pick}}_{v,r,j} + \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$;
          // Update remaining availability.
          $D'^{\text{prod}}_{r,j} \leftarrow D'^{\text{prod}}_{r,j} - \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$;
        **end if**
      **end while**
    **end if**
    // Assign the $x^{t_v}_{v,i,j}$ values only if empty pallets are delivered to suppliers
    **if** $(Q^{\text{empty}}_{v,j,s} == 0)$ **then**
      $i \leftarrow myPrevNode_v$; // Set the previous node for Step 3.
      $t_v \leftarrow \max\{(t_v + \tau_{vij}),\ \min\{T_j\}\}$; // Appropriate time for visiting this
      supplier
      Set $x^{t_v}_{v,i,j} = 1$; // Assign $v$ to visit supplier at appropriate time.
      $myPrevNode_v \leftarrow j$; // Set the previous node for Step 3.
    **end if**
  **end for**
**end if**
```

If there are empty pallets in the vehicle that are not delivered to suppliers due to time window constraints, then the $Q^{\text{empty}}_{v,r,s}$ value is reduced from the LQ_v value. These values need not be added back to $A'^{\text{pal}}_{r,s}$ values, because, if the empty pallets are not delivered in this cycle, they cannot be delivered in future cycles because the upper limit of the time interval is already crossed.

```
// Assign $Q^{\text{empty}}_{v,j,s}$ values to zero.
for all $s \in S''$ do
  for all $r \in R$ do
    // Checks for undelivered empty pallets
    if $(Q^{\text{empty}}_{v,r,s} > 0)$ then
      $LQ_v = LQ_v - Q^{\text{empty}}_{v,r,s}$; // update the loaded quantity
      $Q^{\text{empty}}_{v,r,s} = 0$; // Assign empty pallet quantity to zero
    end if
  end for
end for
// Pick up full pallets from suppliers in set $S'''$.
// Choose the eligible supplier that can be visited earliest from set $S'''$.
// Break ties by selecting the eligible retailer with the smallest ID:
    $j \leftarrow \min\left\{ s \in S'' : \min\{T_s\} = \min_{k \in S'''}\{\min\{T_k\}\} \right\}$;
    $t'_v = \max\{(t_v + \tau_{vij}), \min\{T_j\}\}$;
if $(t'_v \leq \max\{T_j\})$ then
  for all $r \in R$ do
    if $(t'_v \leq \max\{T_r\})$ then
      while $LQ_v < c_v$ do
        // Update loaded quantity on vehicle $v$.
        $LQ_v \leftarrow LQ_v + \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$;
        // Update quantity picked up.
        $Q^{\text{pick}}_{v,r,j} \leftarrow Q^{\text{pick}}_{v,r,j} + \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$;
        // Update remaining availability.
        $D'^{\text{prod}}_{r,j} \leftarrow D'^{\text{prod}}_{r,j} - \min\{D'^{\text{prod}}_{r,j}, c_v - LQ_v\}$;
```

```
          end while
        end if
        // Assign the $x^{t_v}_{v,i,j}$ values only if full pallets are picked up
        if ($Q^{\text{pick}}_{v,r,j} > 0, j \in S'''$) then
          $i \leftarrow myPrevNode_v$; // Set the previous node for Step 3.
          $t_v \leftarrow \max\{(t_v + \tau_{vij}),\ \min\{T_j\}\}$; // Coordinated time for visiting this
          supplier
          Set $x^{t_v}_{v,i,j} = 1$; // Assign $v$ to visit supplier at appropriate time.
          $myPrevNode_v \leftarrow j$; // Set the previous node for Step 4.
        end if
      end for
    end if
  end if
end for
```

Step 4. In this step, each vehicle in set DV is routed to either a Δ^1 node or the nearest CD^{in}. If $LQ_v = 0$, the vehicle is routed to Δ^1. If $LQ_v > 0$, the vehicle is routed to CD^{in}. In cases where the vehicle is routed to CD^{in}, after delivering goods at CD^{in}, the vehicles are routed to the appropriate node in set Δ^1. Each vehicle ends its route at the Δ^1 node, where it is assigned a new vehicle number.

```
for all $v \in DV$ do
  if ($LQ_v = 0$) then
    $i \leftarrow myPrevNode_v$; // Store the previous node for Step 4.
    // Find nearest node, $j \in \Delta^1$, such that $\tau_{v,i,j} = \min_{k \in \Delta^1} \{\tau_{v,i,k}\}$.
    $t_v \leftarrow \{t_v + \tau_{v,i,j}\}$; // Update the latest time a vehicle is assigned to this $\Delta^1$
    node.
    Set $x^{t_v}_{v,i,j} = 1$; // Assign $v$ to visit cross dock at appropriate time.
    $v' \leftarrow$ Next copy of vehicle $v$;
    $t_{v'} = t_v$; // Initialize the starting time of vehicle $v'$.
    $myPrevNode_{v'} = \Delta_j^1$; // Initialize starting node of $v'$.
end if
if $(LQ_v > 0)$ then
    // Find nearest cross dock $j \in CD^{\text{in}}$ such that $\tau_{v,i,j} = \min_{k \in CD^{\text{in}}} \{\tau_{v,i,k}\}$.
    $myCD_v \leftarrow j$; // Store the node number of the chosen CD for Step 4.
    $myPrevNode_v \leftarrow i$; // Store the previous node for Step 4.
    $t_j^{\max} \leftarrow \max\{t_j^{\max}, t_v + \tau_{v,i,j}\}$; // Update latest time a vehicle is assigned to this
    CD.
    // At this point, we have determined the earliest possible time that each
    "pickup" vehicle may be routed to a cross dock. Now, we must coordinate
    the arrival times of these vehicles at each cross dock.
    $Q_{j,r,s}^{\text{in}} = 0 \ \forall \ j \in CD^{\text{in}}, r \in R, s \in S$; // Initialize the quantity of goods arriving
    at the cross dock.
    $i \leftarrow myPrevNode_v$; // Last supplier visited.
    $j \leftarrow myCD_v$; // Selected cross dock.
    $t_v \leftarrow t_j^{\max}$; // Coordinated time for visiting this cross dock.
    Set $x_{v,i,j}^{t_v} = 1$; // Assign $v$ to visit cross dock at appropriate time.
    // In a cross dock, no storage of pallets are allowed, so all the pallets brought by
    the inbound trucks should be transferred to outbound trucks. Thus, the total
    quantity of pallets brought by the inbound trucks should not exceed the total
    capacity of the outbound trucks at a cross dock. To satisfy this condition, the
    heuristic adds the difference in quantity to $D_{r,j}'^{\text{prod}}$.
    $Q_j^{\text{CDin}} = 0 \ \forall \ j \in CD^{\text{in}}$; // Initialize the quantity of goods in the cross dock in.
    $Q_j^{\text{CDout}} = 0 \ \forall \ j \in CD^{\text{out}}$; // Initialize the quantity of goods in the cross dock
    out.
    for all $v \in DV$ do
```

```
        for all $v' \in DV2$ do
            if (v and v', are the $n^{th}$ vehicle in the sequence) then
                $Q_j^{\text{CDin}} = Q_j^{\text{CDin}} + LQ_v$; // Update quantity in $CD^{\text{in}}$
                $Q_j^{\text{CDout}} = Q_j^{\text{CDout}} + C_{v'}$; // Update available capacity at $CD^{\text{out}}$
            end if
        end for
    end for
    for all $v \in DV$ do
        for all $j \in CD^{\text{in}}$ do
            while $((Q_j^{\text{CDin}} > Q_j^{\text{CDout}}))$ do
                // Choose the eligible supplier to be visited
                // Break ties by selecting the eligible supplier with the largest ID:
                    $a \leftarrow \max\left\{s \in S : \min\{T_s\} = \max_{k \in S}\{\min\{T_k\}\}\right\}$;
                // Choose the eligible retailer to be visited
                // Break ties by selecting the eligible retailer with the largest ID:
                    $b \leftarrow \max\left\{r \in R : \min\{T_r\} = \max_{n \in R}\{\min\{T_n\}\}\right\}$;
                if $((Q_j^{\text{CDin}} - Q_j^{\text{CDout}}) \geq Q_{v,b,a}^{\text{pick}})$ then
                    // If full pallets picked up at a supplier for a retailer is less than the
                    quantity to be transferred, then
                    $D_{b,a}'^{\text{prod}} \leftarrow D_{b,a}'^{\text{prod}} + Q_{v,b,a}^{\text{pick}}$; // Update the $D_{b,a}'^{\text{prod}}$ value
                    $Q_j^{\text{CDin}} \leftarrow Q_j^{\text{CDin}}$ - $Q_{v,b,a}^{\text{pick}}$ ; // Update full pallet capacity at $CD^{\text{in}}$
                    $Q_{v,b,a}^{\text{pick}}$ =0; // Update full pallet quantity picked up.
                else
                    $D_{b,a}'^{\text{prod}} \leftarrow D_{b,a}'^{\text{prod}} + (Q_j^{\text{CDin}}$ - $Q_j^{\text{CDout}})$; // Update the $D_{b,a}'^{\text{prod}}$ value
                    $Q_j^{\text{CDin}} \leftarrow Q_j^{\text{CDin}}$ - $(Q_j^{\text{CDin}}$ - $Q_j^{\text{CDout}})$ ; // Update the full pallet capacity
                    at $CD^{\text{in}}$
                $Q^{\text{pick}}_{v,b,a} \leftarrow Q^{\text{pick}}_{v,b,a}$ - $(Q^{\text{CDin}}_{j}$ - $Q^{\text{CDout}}_{j})$ ; // Update the full pallet quantity
                picked up.
              **end if**
            **end while**
          **end for**
        **end for**
        // Update quantity of goods arriving at cross dock
        $Q^{\text{in}}_{j,r,s} = 0 \ \forall \ j \in CD^{\text{in}}, r \in R, s \in S$; // Initialize quantity of goods arriving at
        the CD.
        **for all** $r \in R$ **do**
          **for all** $s \in S$ **do**
            $Q^{\text{in}}_{j,r,s} \leftarrow Q^{\text{in}}_{j,r,s} + Q^{\text{pick}}_{v,r,s}$; // Update quantity of goods arriving at cross dock.
          **end for**
        **end for**
        // Route $v$ to the appropriate $\Delta^1$ node, ending its tour:
            $t_v \leftarrow t_v + \tau_{v,j,\Delta^1_j}$; // Updated time.
            Set $x^{t_v}_{v,j,\Delta^1_j} = 1$; // Assign $v$ to final node.
        $v' \leftarrow$ next copy of vehicle $v$;
        $t_{v'} = t_v$; // Initialize the starting time of vehicle $v'$.
        $myPrevNode_{v'} = \Delta^1_j$; // Initialize starting node of $v'$.
      **end if**
      // Add the vehicle copy to the set DV2.
      set $DV2 \leftarrow v'$; // Update set $DV2$, to be used in VehicleCopy function
    **end for**
```

6.3 The `VehicleCopy` function

Step 1. In this step, first the contents of set DV are updated with the values in set $DV1$. Next, the contents of set $DV1$ are replaced by set $DV2$. The function `DeliveryAndPickup` is repeated with the updated DV. This process continues until all vehicle copies are used. Let *numCopy* be the number of vehicle copies and *numCycle* be the number of cycles that function `DeliveryAndPickup` should be repeated.

```
$numCycle = numCopy * 2$;
for all $numCycle$ do
    // Transfer the contents of set $DV1$ to set $DV$
    $DV \leftarrow DV1$
    // Transfer the contents of set $DV2$ to set $DV1$
    $DV1 \leftarrow DV2$
    Execute function DeliveryAndPickup;
end for
```

The idea behind the development of this heuristic is the expectation that, if the x^t_{vij} values for each vehicle are added as constraints to the existing model, then CPLEX would determine the pick up and delivery quantities for vehicles at the retailer and supplier nodes in the network. The initial solution heuristic can be improved by using metaheuristic approaches, such as tabu search, to select among candidate routes for each vehicle.

Chapter 7

Numerical Analysis

This chapter contains a numerical analysis on generated test problems for the mathematical model. First, analysis is performed for test problems solved via CPLEX. Next, analysis is conducted on test problems solved using CPLEX with the incorporation of the heuristic proposed in the previous chapter. In addition, a procedure to develop test problems is discussed. Numerical analysis helps to identify the time taken to solve the problem, to compare the solutions generated using the heuristic against CPLEX, and to suggest possible improvements to be incorporated in future research.

7.1 Generation of Test Problems

The examples provided in the literature do not provide sufficient numerical values for all of the input parameters required for this thesis work. Therefore, the input parameters are programmed to generate randomly using the software Octave, version 3.2.4. Table 7.1 shows how input parameter values are created.

Parameter	**Value**
$\|R\|$	Hard coded
$\|S\|$	Hard coded
$\|CD\|$	Hard coded
$\|V'\|$	Hard coded
$\|V''_v\|$, where $v \in V'$	Hard coded
α	Hard coded
C_v	U(24,30)
Speed of V	U(50,70) [miles/hr]
D_{rs}^{prod}	U(l,u)
A_{rs}^{pal}	U(0,40)
x coordinate for nodes	U(0,500)[miles]
y coordinate for nodes	U(0,500)[miles]
τ_{vij}	Programmed to calculate

Table 7.1: Input parameters

The parameter values for the number of retailers, suppliers, cross docks, vehicles, number of vehicle copies, and α are hard-coded in the program in order to test the model under different scenarios. The capacity and speed of each vehicle is randomly generated and follow a uniform distribution. The demand of full pallets, D_{rs}^{prod}, follows a uniform distribution, U(l,u), where the lower limit (l) is assumed to be 0 and the upper limit (u) is calculated using the following expression. In the expression, UDemand represents the total demand value at the upper limit:

$$\text{UDemand} = (\text{\# of vehicle copies}) * \sum_{v \in V'} C_v,$$

$$\text{Upper limit} = \text{UDemand}/(\text{\# of retailers} * \text{\# of suppliers}).$$

The demand of empty pallets A_{rs}^{pal}, is randomly generated for each supplier/retailer combination and follows a uniform distribution. The location of the nodes in the network are represented as (x, y) coordinates. Depending on the number of nodes in the network, separate (x, y) coordinates are generated randomly for each node and follow a uniform distribution. With the help of the coordinates, the distance between two nodes in the network is calculated. The speed of each vehicle is randomly generated and the average speed of all vehicles is assumed to have a lower limit of 50 [miles/hr] and an upper limit of 70 [miles/hr], and follows a uniform distribution. With the two data (i.e., the distance between two nodes and average speed of the vehicle), τ_{vij} values, which represent the travel time by vehicle v from node i to destination node j, are calculated.

Table 7.2 shows how the lower limit and the upper limit for the time intervals are randomly generated. One time interval is assumed to be 15 minutes. (i.e., for one hour, four time intervals are considered). The upper limit for the suppliers are considered higher than the retailers to facilitate the back haul operation.

Parameter	**Time Interval Lower Limit**	**Time Interval Upper Limit**
Retailer	U(5,7)	U(16,18)
Supplier	U(4,6)	U(21,23)
Cross Dock	1	24
Δ^1	1	24

Table 7.2: Time interval T_j

All the parameter values are randomly generated only for the calculation purposes. Any user can vary the lower and upper limits and the hard coded values to suit a particular scenario.

7.2 Initial Solution Scheme

Using the aforementioned problem generator, 30 “small” test problems were created for problem sizes shown in Table 7.3, where $| \cdot |$ represents the cardinality of the given set. Ten problems were generated for each row of this table.

$\|R\|$	$\|S\|$	$\|V\|$	Copies	$\|CD\|$	Average # of Decision Variables	Average # of Constraints
5	3	3	1	1	77,486	22,190
10	6	5	1	1	281,044	67,494
20	10	10	1	2	3,130,490	*

Table 7.3: Summary of small problem sizes

CPLEX was unable to find a feasible solution to any of the 30 problem instances within 30 minutes. The reason might be due to the large number of decision variables and constraints. In Table 7.3, columns 6 and 7 give the average number of decision variables and constraints developed for the test problems. It can be inferred that, as the size of the test problem increases, the number of decision variables and constraints also increases drastically.

On average, approximately 3 million decision variables were required for problems with 20 retailers. CPLEX, in the process of solving the problem with 20 retailers, encountered computer memory issues. One reason for this computer memory issue might be due to large number of decision variables. Because of this memory issue, CPLEX was not able to generate the constraints required to find a solution to 20-retailer problems.

Next, the problems were tested with the heuristic explained in Chapter 6. Although the proposed heuristic cannot solve the computer memory issues for larger problems, it was expected to give a solution to smaller problems.

					Average # of	Average # of	Average
$\lvert R\rvert$	$\lvert S\rvert$	$\lvert V\rvert$	Copies	$\lvert CD\rvert$	Decision Variables	Constraints	Run Time [sec]
5	3	4	1	1	77,486	22,190	5.923
10	6	4	1	1	281,044	67,494	37.395
20	10	10	1	2	3,130,490	*	*

Table 7.4: Summary of heuristic solution

Table 7.4 shows the summary of results for the problems solved with the help of the proposed heuristic. Using the heuristic, CPLEX was able to find the solution for test problems with 5 and 10 retailers. Column 8 in Table 7.4 shows the average time taken to solve the problem. An average time of approximately 6 seconds to solve problems with 5 retailers is found to be reasonable. However, for problems with 10 retailers, the average run time of 37 seconds is very high. Because the solution developed by the heuristic is only an initial solution, in future research, this solution should be used in a metaheuristic approach to develop better solutions. For every iteration in a metaheuristic approach, CPLEX should be used to check whether the solution developed is a better solution. For example, if a metaheuristic approach performs 100 iterations, then the solution time will be 3700 seconds, which will be approximately 1 hour. For larger problems, the solution time may increase significantly.

The results for all the individual problems of 5 and 10 retailers are shown in Tables 7.5 and 7.6. For some problems it can be noted that the size of the objective function value is high. For these problems we may infer that some of the nodes are not visited within the given time windows. To reduce the number of undelivered full pallets and empty pallets to their destination nodes, one solution is to increase the number of original vehicles in the system. Column 6 refers to the average number of time intervals considered in each node. Column 7 gives the details of average time window size in percentage. It can be inferred that, for 5-retailer problems, on average, each node is open for delivery or pick up 67% of the available time. For 10-retailer problems, this figure is 61%. The average number of time intervals is calculated by taking into consideration retailer, supplier, CD^{in}, CD^{out}, and Δ^1

nodes. The CD and Δ^1 nodes are open for almost the entire time period. So, when average time window size is calculated only taking into consideration supplier and retailer node, the average time window size is only 53%.

Test Problem #	Objective Function Value	Solution Time [sec]	# of decision Variables	# of Constraints	Avg. # of Time Int.	Avg. Time Window Size [%]
1	6165	6.681834	77474	22198	63.6363	66.28
2	174	4.680644	76834	22070	70.5454	73.48
3	210	5.122504	76322	22006	62.5454	65.15
4	3323	5.83187	78434	22324	64.3636	67.05
5	10136	5.964095	77474	22197	63.6363	66.28
6	112	7.498031	78818	22389	64.7272	67.42
7	10957	6.281562	77026	22133	63.2727	65.90
8	10517	5.045159	75874	21941	62.1818	64.77
9	176	5.550253	77026	22133	63.2727	65.90
10	2122	6.573162	79586	22516	65.4545	68.18

Table 7.5: Details of heuristic performance on problems with 5 retailers

Test Problem #	Objective Function Value	Solution Time [sec]	# of decision Variables	# of Constraints	Avg. # of Time Int.	Avg. Time Window Size [%]
1	8803	72.470761	279562	67363	58.9474	61.40
2	14713	31.846418	277098	67167	58.3157	60.74
3	15047	27.31317	280042	67424	59.1578	61.22
4	15831	32.395646	282634	67617	59.7894	62.28
5	13824	34.230393	283370	67683	60.0000	62.50
6	17061	32.975578	280906	67484	59.3684	61.84
7	17847	34.551907	284714	67806	60.4210	62.93
8	11117	31.887243	281034	67488	59.3684	61.84
9	17070	44.365912	280042	67419	59.1578	61.62
10	15009	31.912833	281034	67490	59.3684	61.84

Table 7.6: Details of heuristic performance on problems with 10 retailers

Two main problems that were experienced in this modeling approach are computer memory issues and long runtimes to solve the problem. Some of the recommendations to address these issues are as follows. First, in order to solve the problem related to computer memory issues, the mathematical problem must be modeled with fewer decision variables. If the number of decision variables is reduced, the memory space consumed may also be reduced. Second, the problem should have narrow time windows. For the test problems, the average time window size considered is approximately 63%. If the time windows are narrowed, then the number of decision variables in the model will get reduced. Third, the current model has a time slice for every 15 minutes. If the time slice is increased (i.e, a time slice for every 30 minutes or 1 hour), then the number of decision variables may also be reduced. Fewer decision variables may in turn reduce the time taken to solve the problem.

From this numerical analysis, it can be concluded that a more efficient mathematical model must be developed with fewer decision variables. This may help to overcome the computer memory problems, and may reduce the time taken to solve the problem.

Chapter 8

Conclusions & Future Research

8.1 Conclusions

The research work tries to integrate the vehicle routing problem in a cross docking environment, to provide better utilization of the vehicle fleet. This research introduces new formulations to solve vehicle routing problem with pick up during the linehaul operation and delivery during the backhaul operation. The entire problem is based on a cross docking environment with allowable split deliveries and time window constraints using a heterogeneous fleet. The research work carried out to date by various authors addresses the VRP model with cross docks with only a subset of the different characteristics mentioned above. In this research work the mathematical model is developed using all of the above mentioned characteristics. The goal of this research work is to build a mathematical model with the above mentioned criteria and to produce a feasible solution.

The problem is modeled as a mixed integer linear programming (MILP) model. The model was validated using CPLEX. During the validation stage it was found that CPLEX was not able to solve the problem within the test time of 30 minutes. However, it was found that if the schedules were modeled as constraints to the model, CPELX was able to find a feasible solution to the problem (although not optimal) and it was able to determine the quantity of full and empty pallets that must be transferred from one node to another. This motivated the need to develop a heuristic solution approach. The goal of the heuristic was to determine feasible routes and schedules for every vehicle in the system. The output of the heuristic is a collection of x_{vij}^t values, where i is the starting node and j is the destination node of vehicle v, and t is the time the vehicle reaches node j. These x_{vij}^t values are treated as constraints of the model, in addition to the already existing node and network constraints.

A sample problem was generated as an example to demonstrate the different features in the model. A procedure to develop the test problems is described in this thesis, as well. The existing literature does not provide test problems with numerical values for all of the parameter values described in this model, thus forcing the development of a procedure to generate test problems with maximum flexibility.

Numerical analysis was conducted on three sets of test problems with retailer node sizes of 5, 10 and 20. Each set contained 10 test problems, totalling 30 test problems. All of the test problems were first solved using CPLEX with a maximum run time of 30 minutes. CPLEX was not able to find a feasible solution. Next, all of the problems were solved using CPLEX with the help of the heuristic and CPLEX was able to find solutions for problem sizes of 5 and 10 retailers. During this stage computer memory issues were encountered when large problems were solved. Recommendations for overcoming the computer memory issues were also discussed.

At the end of the numerical analysis, it was concluded that the present mathematical model and the developed heuristic work well for the small problem sizes, but there is a need for a better model with fewer decision variables and time intervals to solve larger problems.

8.2 Future Research

The research work can be extended by incorporating metaheuristic approaches to solve the problem, and by developing a mathematical model with fewer decision variables. Instead of empty pallets that are considered in this model, a more realistic reverse logistic operation of transporting returned or damaged products back to the supplier for service or replacement can be considered.

Currently, constant transfer time is assumed for transferring pallets at the cross dock, regardless of the quantity of pallets transferred or the location of the inbound and outbound trucks. In future research, more realistic transfer time of pallets can be introduced to the model depending on the number of pallets to be transferred and travel time of pallets from the inbound to outbound doors of the cross dock. Moreover, the model can be improved by including a separate optimization procedure to assign trucks to the cross dock to reduce the travel time of pallets inside the cross dock.

Currently, by assumption, there is no space to store inventory in the cross dock and pallets must be transferred directly from the inbound trucks to the outbound trucks. To satisfy this condition, a group of trucks are scheduled for simultaneous arrival and departure. This constraint can be improved by allowing storage of inventory at the cross dock. This requires constraints that limit the quantity of goods in inventory and also limit the length of time that these goods remain in the cross dock. This also requires modifying the current model to allow transfer of goods without requiring all trucks to be at the cross dock simultaneously. As a result, vehicle utilization may be improved.

Bibliography

U.M. Apte and S. Viswanathan. Effective cross docking for improving distribution efficiencies. *International Journal of Logistics Research and Applications*, 3(3):291–302, 2000. ISSN 1367-5567.

Nils Boysen and Malte Fliedner. Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413 – 422, 2010. ISSN 0305-0483.

P. Chen, Y. Guo, A. Lim, and B. Rodrigues. Multiple crossdocks with inventory and time windows. *Computers & operations research*, 33(1):43–63, 2006.

S.M. Disney, A.T. Potter, and B.M. Gardner. The impact of vendor managed inventory on transport operations. *Transportation Research Part E: Logistics and Transportation Review*, 39(5):363–380, 2003. ISSN 1366-5545.

Y. Dong and K. Xu. A supply chain model of vendor managed inventory. *Transportation Research Part E: Logistics and Transportation Review*, 38(2):75–95, 2002.

D. Gulczynski, B. Golden, and E. Wasil. The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering*, 2011.

M. Gumus and J.H. Bookbinder. Cross-docking and its implications in location-distribution systems. *Journal of Business Logistics*, 25(2):199–228, 2004.

M. Hammer. Deep change. *Harv Bus Rev*, 82:84–93, 2004.

J.C. Johnson and D.F. Wood. *Contemporary logistics*. The Macmillan series in marketing. Macmillan, 1990. ISBN 9780023608414.

R. Kaipia, J. Holmstrom, and K. Tanskanen. Vmi: What are you losing if you let your customer place orders? *Production Planning & Control*, 13(1):17–25, 2002.

V.B. Kreng and F.T. Chen. The benefits of a cross-docking delivery strategy: a supply chain collaboration approach. *Production Planning & Control*, 19(3):229–241, 2008. ISSN 0953-7287.

Y.H. Lee, J.W. Jung, and K.M. Lee. Vehicle routing scheduling for cross-docking in the supply chain. *Computers & Industrial Engineering*, 51(2):247–256, 2006. ISSN 0360-8352.

C.J. Liao, Y. Lin, and S.C. Shih. Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications*, 37(10):6868–6873, 2010. ISSN 0957-4174.

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

H. Ma, Z. Miao, A. Lim, and B. Rodrigues. Crossdocking distribution networks with setup cost and time window constraint. *Omega*, 39(1):64–72, 2011.

Z. Miao, F. Yang, K. Fu, and D. Xu. Transshipment service through crossdocks with both soft and hard time windows. *Annals of Operations Research*, pages 1–27, 2010.

A. Mingozzi, S. Giorgi, and R. Baldacci. An exact method for the vehicle routing problem with backhauls. *Transportation Science*, 33:315–329, 1999. ISSN 0041-1655.

C.C. Murray and M.H. Karwan. An extensible modeling framework for dynamic reassignment and rerouting in cooperative airborne operations. *Naval Research Logistics (NRL)*, 2010.

R. Musa, J.P. Arnaout, and H. Jung. Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. *Computers & Industrial Engineering*, 59(1): 85–92, 2010. ISSN 0360-8352.

J. Privé, J. Renaud, F. Boctor, and G. Laporte. Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of the Operational Research Society*, 57(9):1045–1052, 2005. ISSN 0160-5682.

R. Sabath. Volatile demand calls for quick response: the integrated supply chain. *Logistics Information Management*, 8(2):49–52, 1995.

R. Soltani and S.J. Sadjadi. Scheduling trucks in cross-docking systems: A robust meta-heuristics approach. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):650–666, 2010. ISSN 1366-5545.

CS Sung and SH Song. Integrated service network design for a cross-docking supply chain network. *Journal of the Operational Research Society*, 54(12):1283–1295, 2003.

P. Toth and D. Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31:372–3852, 1997. ISSN 0041-1655.

M.A. Waller, C.R. Cassady, and J. Ozment. Impact of cross-docking on inventory in a decentralized retail supply chain. *Transportation Research Part E: Logistics and Transportation Review*, 42 (5):359–382, 2006. ISSN 1366-5545.

M. Wen, J. Larsen, J. Clausen, J.F. Cordeau, and G. Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60(12):1708–1718, 2008. ISSN 0160-5682.

R.A. Wilson. *Transportation in America.* Eno Transportation Foundation, 19 edition, 2002.

C.A. Yano, T.J. Chan, L.K. Richter, T. Cutler, K.G. Murty, and D. McGettigan. Vehicle routing at quality stores. *Interfaces*, pages 52–63, 1987. ISSN 0092-2102.

Y. Zhong and E.H. Aghezzaf. Combining dc-programming and steepest-descent to solve the single-vehicle inventory routing problem. *Computers & Industrial Engineering*, 2011.