Collaborative Logistics in Vehicle Routing

by

Selvaprabu Nadarajah

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2008

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Less-Than-Truckload (LTL) carriers generally serve geographical regions that are more localized than the inter-city routes served by truckload carriers. That localization can lead to urban freight transportation routes that overlap. If trucks are travelling with less than full loads there may exist opportunities for carriers to *collaborate* over such routes. That is, Carrier A will also deliver one or more shipments of Carrier B. This will improve vehicle asset utilization and reduce asset-repositioning costs, and may also lead to reduced congestion and pollution in cities. We refer to the above coordination as "collaborative routing". In our framework for collaboration, we also propose that carriers exchange goods at logistics platforms located at the entry point to a city. This is referred to as "entry-point collaboration".

One difficulty in collaboration is the lack of facilities to allow transfer of goods between carriers. We highlight that the reduction in pollution and congestion under our proposed framework will give the city government an incentive to support these initiatives by providing facilities. Further, our analysis has shown that contrary to the poor benefits reported by previous work on vehicle routing with transshipment, strategic location of transshipment facilities in urban areas may solve this problem and lead to large cost savings from transfer of loads between carriers.

We also present a novel *integrated* three-phase solution method. Our first phase uses either a modified tabu search, or a guided local search, to solve the vehicle routing problems with time windows that result from entry-point collaboration. The preceding methods use a constraint-programming engine for feasibility checks. The second phase uses a quad-tree search to locate facilities. Quad-tree search methods are popular in computer graphics, and for grid generation in fluid simulation. These methods are known to be efficient in partitioning a two-dimensional space

for storage and computation. We use this efficiency to search a two-dimensional region and locate possible transshipment facilities.

In phase three, we employ an *integrated* greedy local search method to build collaborative routes, using three new *transshipment-specific* moves for neighborhood definition. We utilize an optimization module within local search to combine multiple moves at each iteration, thereby taking efficient advantage of information from neighborhood exploration. Extensive computational tests are done on random data sets which represent a city such as Toronto. Sensitivity analysis is performed on important parameters to characterize the situations when collaboration will be beneficial. Overall results show that our proposal for collaboration leads to 12% and 15% decrease in route distance and time, respectively. Average asset utilization is seen to increase by about 5% as well.

Acknowledgements

I would like to express my gratitude to Prof. Jim Bookbinder for being a truly exceptional supervisor and mentor throughout my masters program.

Comments from my readers Prof. Peter Van Beek, Prof. Jim Bookbinder and Prof. Samir Elhedli were very helpful in refining my thesis. I thank them for their time. A special thanks to Prof. Van Beek. His courses on artificial intelligence and constraint programming inspired many ideas used in this thesis.

Another special mention goes out to my team members, Jim Bunker and Nourredine Hail in the Optimization and Modelling group at the Canadian Tire Corporation. I thank them for their understanding and willingness to work around my schedule during the last four months of my thesis.

I must also acknowledge my friends Emre Çelebi, Bev Rodgers, Ali Ülku, and Navneet Vidyarthi for their love and care. Finally, I would like to thank my family for their continued support and encouragement.

Dedication

This thesis is dedicated to my parents Nadarajah Angappan and Padmaawathy Nadarajah, who have motivated and inspired me in many ways.

This thesis is also dedicated to my sister, Sooganthy Nadarajah for always encouraging me to be ambitious.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optimization of supply chain and logistics operations has received attention from industry and academia alike over the last few decades. More recently, competitive pressures, economic volatility and increased service expectations have forced companies to look outside their own operations. By sharing information with potential *competitors*, and optimizing joint operations, companies try to eliminate costs that cannot be individually controlled. Collaboration between less-than-truckload (LTL) carriers provides such an opportunity, since local intra-city trucking costs are a staggering annual US $ 435 billion (Wilson [50]). This high cost also means that small improvements in operations will result in large cost savings.

Together with the greater service expectations and competition, there has also been increasing concerns regarding pollution levels in urban regions. To make things worse, urbanization has increased the volume of truck traffic in cities, which has led to congestion problems among others. The cost of congestion caused by trucking in Toronto and Peel regions was an annual US $ 2 billion in 1987 (Taylor [45]), and this figure would have increased to a much larger value today. Commercial trucking in cities has been identified as a major contributor to both the nuisances of congestion and pollution.

This thesis provides a *collaborative framework* for LTL carriers, where joint optimization of operations through collaboration will lead to savings in cost, and at the same time, hopefully curb the negative effects of trucking on the environment and the city. Understandably, trucking operations may now be more complicated, but companies may accept this proposal if the accompanying cost savings are large. As an added bonus, those firms will be identified as environmental stewards. Further, benefits to the city from carrier collaboration may give incentives for the city government to support that initiative. *The main goal of this thesis is to show that our proposed framework for carrier collaboration can lead to the preceding benefits.*

The thesis thus has two main contributions. Firstly, we define a new framework for collaboration between LTL carriers that contains two stages. The first stage involves exchange of (partial) loads between carriers at the entry to the city, while trucks make such exchanges during local delivery in the second stage. We also explain that carrier collaboration needs to be studied independently of *shipper* collaboration. We will define the latter, and show that the benefits differ in the two cases.

Our second contribution is an *integrated* three-phase heuristic to solve the mathematically complicated problem that results from our two stage collaborative framework. In the first phase, we use an integrated version of tabu search, or of guided local search, to solve vehicle routing problems with time windows (VRPTWs). This method uses a constraint-programming engine to check for feasibility and reduce the search space.

In the second phase, we use an *adaptive quadtree search* method, which is popular in grid generation for fluid simulations and for data storage in image processing. The quadtree search is used to efficiently explore the two-dimensional region which represents the city, and to create clusters of customers that can be considered for collaborative exchange of partial loads at transshipment points. The site of the

transshipment point is also located in the cluster by the preceding method. In the last phase, we use an integrated greedy local search method to construct collaborative routes. This time, the integration is between heuristics and optimization.

To our knowledge, no previous work on the same problem exists. Therefore, we also present a method to create random data sets for the carrier collaboration problem. This is an extension of the method used to create random VRPTW data sets in Solomon [41]. We perform extensive computational tests to show that collaboration does indeed provide the benefits that we claim above.

The thesis is organized as follows. The next chapter provides the necessary background for understanding the problem and solution methodology. In Chapter 3, we present details of the proposed collaborative framework, some geometric proofs, and the three-phase heuristic. Results from extensive computational testing and evaluation are provided in Chapter 4. The final chapter provides a summary, concluding remarks, and extensions for future research.

Chapter 2

Background

This chapter introduces the required background knowledge to understand the remaining chapters of this thesis. It divides into subsections, which relate to collaborative logistics, the vehicle routing problem, constraint programming, quadtree search and local search. Whereever exhaustive details are important, a suitable reference will be given to guide the reader.

2.1 Collaboration in Carrier Logistics

The business landscape is constantly evolving and the internet has fostered opportunities that were once not available. Cooperation has become a buzzword in industry; the internet serves as an ideal platform to nurture these cooperative initiatives.

Though cooperation involves interaction between companies, it does not derive the complete benefits that can be achieved through collaboration. Collaboration indicates a stronger relationship, between firms or supply chains, than cooperation. We broadly define collaboration as: "The coordinated flow of material and information within and between supply chains' vertical and horizontal structure.

The goal is to mutually improve the efficiency of the supply chains locally, among collaborating members, and globally across all supply chains involved."

The key to supply chain collaboration is a mutually beneficial outcome for all collaborating members. Still, many are skeptical about the outcome of collaboration. Some suggest that collaboration may lead to a conflict if not managed correctly. Therefore, the Voluntary Interindustry Commerce Standards Association (VICS) developed Collaborative Planning, Forecasting, and Replenishment (CPFR) to establish guidelines. CPFR is a nine-step business process model for value chain partners to coordinate sales forecasting and replenishment in order to reduce variance between supply and demand [1].

A typical supply chain has suppliers, manufacturers, retailers, customers and other third parties. In addition to reduction of costs, Angelides and Angerhofer [2] points out that increasing the number of players of each type in a collaborative supply chain will lead to a reduction in uncertainty for a given length of the chain. This improves the supply chain's competitiveness and is of strategic importance. Those authors also present a framework and a performance-measurement system for collaborative supply chains. The model deals with collaboration at the strategic, managerial and operational levels; at each stage the focus is on measurement of results.

In the preceding paragraphs, we introduced collaboration in supply chain management. In the remaining subsections, we introduce collaborative logistics and operational collaboration in more detail.

2.1.1 Collaborative Logistics

Collaborative logistics (CL) is a recent business model designed to eliminate transportation inefficiencies by taking a holistic view of logistics operations. This also

falls under collaborative supply chain management. Though many business models have the same goal, the key difference is that CL is focused on reducing those costs that cannot be controlled by individual firms. That reduction is achieved through inter-firm collaboration.

CL can be seen as a process which has an overlap with CPFR, but which can also be implemented independently. Similar to CPFR, guidelines are important since it also involves a paradigm shift in viewing competitors as potential collaborators. Collaboration between competitors necessitates a neutral platform, and the pervasiveness of the internet has established a neutral and cost effective channel for such collaborative partnerships. The seven immutable laws which set the ground rules for CL are presented in Langley [28]. Collaborative transportation management (CTM) is often the designation for CL in industry. We retain the use of CL unless a distinction is required.

The search for collaboration in transportation is a direct result of global and local supply chains reacting to competitive pressures. The roots of both CPFR and CL are in Vendor Managed Inventory (VMI). In VMI, the vendor or supplier monitors and controls the decisions related to quantity and timing of orders. VMI involves a collaborative partnership between the supplier and retailer and has been identified with many benefits (Gumus et al. [21], Waller et al. [49]). However, it has two main deficiencies.

Firstly, VMI transfers responsibility to the manufacturer while the retailer still dictates most of the rules, which makes the collaboration ineffective. Secondly, VMI fails to consider the influence of the *carrier*. The benefits of collaboration depend on transportation carriers, who need to be part of the collaborative process to avoid surprises on the timing and sizes of planned shipments. Carrier capacities and transportation lead times can then be aligned with supply chain efficiency.

CPFR and CTM were developed by the VICS to address these two inefficiencies. CTM (Sutherland [42]) was an extension of CPFR to include the carrier as a Supply Chain player, to reduce costs, increase asset utilization, and improve service and revenue. CTM views the shipper, carrier and receiver (consignee) as three principle players in the supply chain. We think the progression from VMI to CTM is almost evolutionary.

The basic principle underlying CL is quite simple. By collaborating with potential competitors, companies are integrating multiple supplier and carrier networks. This allows firms to benefit from expanded opportunities. Collaboration itself is enabled by sharing information and enhanced communication between all "collaborating partners."

This raises two important questions. The first is: How much information should be shared? This depends on the degree of collaboration. Several levels of collaboration are suggested in [42]; the extent of information sharing increases with each level (Figure 2.1). The second question is: How can information sharing be facilitated? It is facilitated by use of a safe and common information hub. Such hubs are usually maintained by 3PLs such as Nistevo or Transplace, who provide confidential and specialized collaborative services. For example, Nistevo [31] was able to identify a particular dedicated continuous move route which resulted in a 19% cost savings for collaborating shippers, in addition to improving their truck utilization and reducing driver turnover.

Another important benefit from CL is improvement in customer service. This is a vital factor for firms to stay competitive. Over time, the service requirements have become more stringent due to internet orders and promised delivery dates. Therefore, transit time uncertainty needs to be reduced to attain the desired service. One consequence of a company's mission to achieve excellent customer service is the resulting increase in transportation cost. In recent years, greater driver turnover

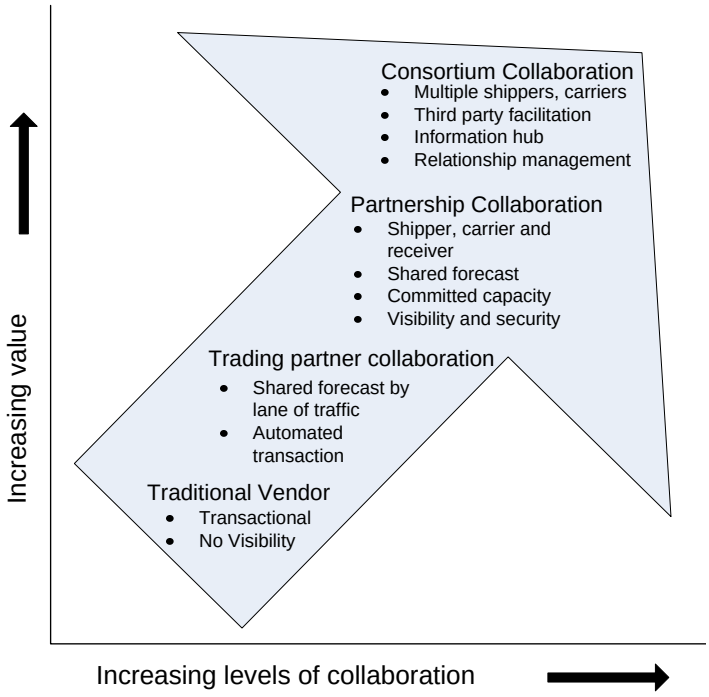


Figure 2.1: Levels of collaboration

and deadhead miles, revised hours of operation and heightened security have all contributed to the soaring cost of transportation. Companies have resorted to CL, working together to eliminate inefficiencies, reduce costs and improve service.

CL also applies across different time horizons, from strategic to operational. Strategic plans concern supply chain network design, fixed asset planning, etc. Tactical-level plans involve collaboration in transportation procurement and contracting. The most dynamic form of CL is operational collaboration. This pertains to enhancing asset utilization through better shipment and carrier management, and improved fleet *routing* and *scheduling*.

Operational collaboration is highly complex and requires information to be shared between competitors. However, as competition forces carriers to reduce costs, they have little choice but to master collaboration. This thesis focuses on Carrier collaboration which falls in the operational category of CL.

2.1.2 Operational Collaboration

There are two types of operational collaboration in CL, namely shipper and carrier collaboration. In shipper collaboration, shippers form communities and collaborate in order to bundle lanes. A lane is a contiguous portion of highway or road, considered by the carrier as a single link for routing purposes.

Carriers prefer bundled lanes, as they may lead to what are termed *continuous moves*. Continuous Move Routes (CMR) are ones in which the carrier's truck is always full. Such a route will ideally have zero deadhead miles and no asset-repositioning costs. The latter costs are incurred when a truck travels empty between two stops. Reduction in asset repositioning costs can lead to large savings for carriers, since trucks in the USA travel empty twenty percent of the time on average (Wilson [50]).

This reduction in cost allows carriers to offer more competitive rates to the shipper, thereby providing an incentive for shippers to collaborate. Shipper collaboration also leads to recurring work for drivers, which is important as driver turnover has reached a record high in recent years.

CSCMP's 18th Annual State of Logistics [50] report states that transportation costs made the biggest leap by increasing 9.4% from 2005 to 2006. The cost incurred in 2006 by motor carriers alone was a staggering US $635 billion. This further splits into intercity and local (Figure 2.2) and amounts to $432 billion and $203 billion respectively. Therefore, even a small percentage decrease in cost through collaboration can translate to substantial reductions in *real cost*.

Shipper collaboration enables lower costs because of the bundling of lanes by a single carrier. Still greater benefits could be achieved if there were multiple carriers, and they collaborated. We call this *carrier collaboration/Less-Than-truck Load (LTL) collaboration*. There are two types of carriers, Truckload (TL) and

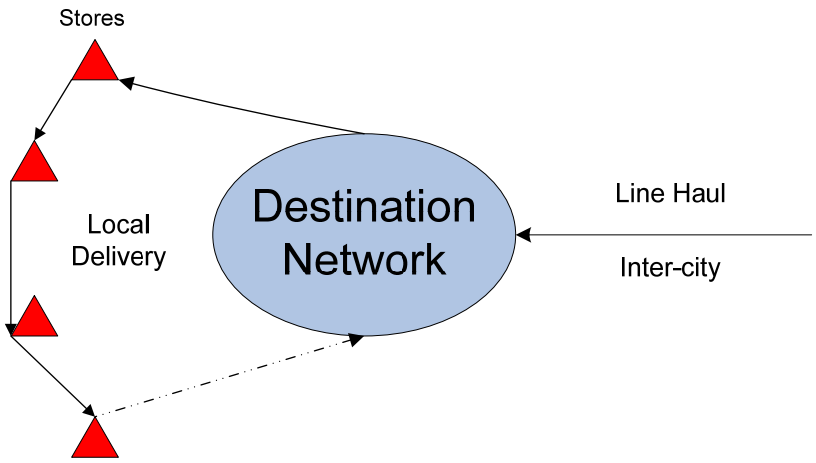


Figure 2.2: Line haul and local delivery

LTL. TL carriers predominantly provide a point-to-point service, similar to the linehaul or intercity route represented in Figure 2.2. In this case, repeatability of routes and continuous moves from the combined lanes are worthwhile.

LTL carriers, on the other hand, are concerned with delivery of small shipments (between 500-15000 lbs on average) , ultimately over a limited geographical region, similar to local delivery shown in Figure 2.2. LTL collaboration aims at designing continuous moves which minimize asset-repositioning cost. As before, the bundling of routes is certainly beneficial, but the loads tendered to LTL carriers are usually small in size and not predictable. Therefore, benefits from shipper collaboration are much reduced for LTL carriers.

As mentioned above, continuous moves have been the focus of many optimization models in transportation (Robin and Levary [35], Desrosiers et al. [13], Savelsbergh [40]). Before proceeding further, we briefly define continuous moves to avoid any confusion. Continuous moves are routes which possess characteristics aimed at increasing truck utilization and taking advantage of the economies of scale from combined loads. Certain restriction may also apply and include limits on dead head miles, total route length, waiting time of trucks between each load delivery or pickup, and minimal distance of a loaded leg. The preceeding list is not exhaustive

and the characteristics vary depending on the application. The definition of continuous moves changes depending on whether it refers to the TL or LTL industry. TL CMs are defined as having a combination of inbound or outbound loads by Ronen [37]. In contrast, LTL CMs consist of inter-woven pick-up and delivery opportunities. An elastic set partitioning problem for the dispatching of orders with different truck modes and route types is also presented in [37]. Unlike many others Ronen [37] makes a strong statement that CMs are only one type of route and should be considered in addition to other route options. We take a similar stance in our research on collaborative routes. This will be re-iterated in later sections. In the remaining part of this section, we review the most relevant literature on operational collaboration.

Shipper collaboration has been the focus of recent academic studies (Ergun et al. [15, 14]). The lane covering problem (LCP) was introduced in [15] and finds a set of minimum cost cycles which cover a given set of lanes. The paper suggests that shippers should collaborate and submit routes to carriers, instead of submitting individual lanes, in return for favorable rates. The authors present a polynomial time algorithm for the unconstrained version of the LCP. They show that the length-constrained LCP (CCLCP) is NP-hard and present a greedy heuristic which they conjecture to be a 1.5 approximation algorithm.

The heuristic in [15] to solve the LCP has a pre-processing stage where feasible cycles are generated. In the second stage, a greedy heuristic is used to find a set of feasible cycles which cover the set of lanes at minimum cost. Their computational results show that more lanes produce better results in terms of solution quality and total routing cost. Further, they perform a trade-off study between generating all cycles and cycles with only one repositioning arc. The results show that the former case can realize total cost reductions of up to 40% compared to the latter case. The solution quality (i.e. ratio of repositioning length to cycle length) is also better.

Though their experiments establish the anticipated fact that more collaborative opportunities result in better collaborative outcomes, they fail to provide sufficient evidence of why shippers must collaborate in the first place.

In their follow up paper [14], they study the time-constrained LCP. They develop a two-phase heuristic for the CCLCP. In phase 1, a heuristic generates a large number of time-feasible cycles and then greedily selects the subset of these cycles which cover the set of lanes. Following this, a local improvement heuristic is used to improve the quality of the phase-1 solution in phase 2. Though cycles are time feasible, optimization is required to choose the starting arc, such that the total cycle duration is minimized. Local improvement merges two cycles by removing the largest repositioning arcs from both cycles and optimally reconnecting them to form another single cycle.

Extensive computational testing is performed by varying R, the maximum travel time of repositioning arcs in a cycle, and the ratio of number of lanes to number of customers (lane-point ratio). They use the percentage of the repositioning distance to total distance, and the percentage of non-lane travel time to total distance as criteria to measure quality. Results show that quality increases with increase in R, and both criteria are lower by 2-3% for higher lane-point ratio. Experiments also show that only a few lanes are present in a cycle, and that the local improvement heuristic performs extremely well (20-50% reduction in cost of phase 1 solution). Further, imposing a supply chain structure reduces the solution quality. The authors suggest that this is due to the absence of incoming arcs to suppliers and outgoing arcs from customers, thereby making natural cycles more difficult in this case. A simplified version of a real-life case showed 5.5 - 13% savings through shipper collaboration.

From the recent literature and the above discussion, it should be clear that shipper collaboration can be highly beneficial. However, carrier collaboration may lead

to benefits different from shipper collaboration. Therefore, that needs to be studied independently. We highlight these benefits in the context of LTL carriers by using simple examples. To the best of our knowledge, carrier collaboration/LTL collaboration has not been studied to date. Defining LTL collaboration and developing a framework for its study, are among the main contributions of this thesis.

2.2 The Vehicle Routing Problem

In this section, variants of the vehicle routing problem relevant to collaborative routing are introduced briefly to familiarize the reader with the important routing characteristics to be discussed in later sections.

Definition 2.2.1. VRP: Given a set of m vehicles with capacities $c_1, .., c_m$ and a set of n customers with known demands, $q_1, .., q_n$, the problem is to solve for the routes of each vehicle starting and ending at a depot, so that the customer demands are served, vehicle capacities are not exceeded in any route, a set of side constraints are satisfied, and an objective is optimized.

2.2.1 Multi-Depot Vehicle Routing Problem (MDVRP)

A company may have several depots from which it can serve its customers. If the customers are clustered around depots, then the distribution problem should be modeled as a set of independent VRPs. However, if the customers and the depots are inter-mingled, then a Multi-Depot Vehicle Routing Problem should be solved. A MDVRP requires the assignment of customers to depots, at each of which a fleet of vehicles is based. Every vehicle originates at the depot, services its assigned customers, and returns to the same depot.The objective of the problem is to service all customers while minimizing the number of vehicles and travel distance.

2.2.2 Pickup and Delivery Vehicle Routing Problems (PDVRP)

In Pick up and Delivery vehicle routing problems each customer is associated with a pair of locations, one location being the pickup point and the other the destination. Additional constraints, known as pairing constraints, need to be imposed to ensure that the same vehicle that picked up the load of a particular customer does the delivery. Further, precedence constraints must be added so that the pickup is done before delivery. Therefore, this problem becomes more difficult to solve, and due to the added restrictions, requires additional vehicles. It is common to use transshipment points in PDVRP so that there can be exchange of loads between the vehicles. Such a problem is referred to as the pickup and delivery vehicle routing problem with transshipment (PDVRPT).

2.2.3 Vehicle Routing Problems with Time Windows (VRPTW)

Intrinsically, the VRP is a spatial problem. During the last few decades, however, temporal aspects of routing problems have become increasingly important. Specific examples of problems with *time windows* include bank deliveries, postal deliveries, industrial refuse collection, school-bus routing, and situations where the customer must provide access, verification, or payment upon delivery of the product or service. Customers in these problems can be served only during certain hours of the day, such as office hours or the hours before the opening of a shop. For example, a warehouse may only accept deliveries within a particular time interval (time window). Therefore, much attention has been given to the Vehicle Routing Problem with Time Windows (VRPTW).

The time windows can be "hard" or "soft". In the hard time-window case, if a vehicle arrives too early at a customer, it is permitted to wait until the customer is ready to begin service. However, a vehicle is not permitted to arrive at a customer

after the latest time to begin service. In contrast, in the soft time-window case, those windows can be violated at a cost. The two variants with multiple pickups or pickups and deliveries discussed above, can be complicated with the addition of time windows. Though time window constraints are not simple, they can help reduce the search space, depending on their tightness.

2.3 Quadtree Search

Tree data structures can be used to efficiently partition complex shapes in a logical and efficient manner. An algorithm which uses a tree data structure for exploration is known as a tree-search algorithm. This section will give an introduction to quadtrees, which are employed in phase 2 of our three-phase heuristic method to solve the problem of carrier collaboration.

Quadtrees were first proposed in Finkel and Bentley [16] for the storage of data with two-dimensional keys. Quadtrees can be classified into region quadtrees and point quadtrees. In region quadtrees, a two dimensional space is recursively decomposed into four smaller quadrants, starting from a bounding rectangle, until some termination criteria ends the recursion. In point quadtrees, the two dimensional point data are stored using quadtrees.

Quadtrees follow a "search-tree" property which allows efficient searching. For example, consider the well known binary tree, where each node n_i has two children and a $key[n_i]$, which is the search key corresponding to node n_i. The binary-search-tree property states that if a node n_k is in the left subtree of n_i, then $key[n_k] \leq key[n_i]$, and $key[n_k] \geq key[n_i]$ if node n_k is in the right subtree of n_i. This property can be used to decide whether the search for a key should proceed in the right or left subtree. Therefore, searching can be done in $O(d)$, where d is the height of the binary tree.

In a quadtree, each internal node has four children. Since the key has two dimensions, a quadtree-search property can be developed. If we assume that the 2-D key represents the coordinates of a region's centroid, then it is logical to partition such that child 1,2,3 and 4 correspond to the NE, NW, SW and SE quadrants respectively. When searching the tree for a particular key, the above property can be used to find the record in $O(d)$, where d is the depth of the quadtree. Again, at each node the above property directs the search to the subtree containing the key, thereby avoiding wasteful search.

Another important property for trees in general is the idea of a *Balanced* Tree: Its leaves must all be at the same depth, resulting in a tree depth of $O(log(n))$. This is important in data storage as highly unbalanced trees in the worst case are linked lists. However, in our application quadtrees are used to decompose regions, and highly unbalanced trees improve the search speed in conjunction with the quadtree-search property. Unbalanced trees are created using termination criteria, which help to reduce the tree's size.

Quadtrees are widely used in computational fluid dynamics to partition complex bodies so that highly adaptive meshes can be generated for simulation. Applications in graphics include storage of data from pictures. In this thesis, an adaptive version of the region quadtree is used. Adaptive quadtrees use multiple termination criteria to reduce the size of the tree and ensure that only relevant information is stored. Consequently, these trees have interesting results concerning computational complexity, which are presented in Appendix 5.3.

We use an adaptive quadtree search to locate transshipment facilities, which are then used to transfer goods between trucks of different carriers. More details can be found in Section 3.5.1.

2.4 Constraint Programming

This section explains constraint programming in a largely non-technical manner. Our goal is to highlight the strengths of constraint programming and the reason for its incorporation in the solution methods to be described later. Since we have not contributed to any new constraint programming technology, this introduction is sufficient to understand the thesis, but the reader is urged to refer to Rossi et al. [38] for further details.

A constraint programming (CP) model is defined by a set of variables $\{x_1, ..., x_n\}$, a finite domain D_i for each variable, and a set of constraints $C_{i_1 \ldots i_k}$ over the variables $x_{i_1}, ..., x_{i_k}$. This is known as a *Constraint Satisfaction Problem (CSP)*. The domain of a variable is the set of all values it can be assigned. A constraint $C_{i_1 \ldots i_k}$ restricts the values that the variables in its definition can take simultaneously. The number of variables over which a constraint is defined is known as the *arity* of a constraint. Constraints of arity two are known as binary constraints.

A CP model of a problem is quite different from its Mathematical Programming (MP) counterpart. Further, CP models are very "natural" and have a rich language with which to model constraints. In contrast, MP constraints are either equalities or inequalities, which are restrictive. As a result, an MP model usually does not perform well in a CP solver. However, it is well known that MP solvers are superior in solving optimization problems which possess certain mathematical structure.

Consider for example the infamous Travelling Salesperson Problem, where the objective is to find the minimum-cost route over a set of visits, where each visit must be performed exactly once. In an MP formulation, we use binary decision variables x_{ij}, where $x_{ij} = 1$ when a visit to j is performed immediately after the visit to i, and 0, otherwise. A CP formulation would use the variables s_i, where the value of s_i would give the immediate successor of visit i. The domain of s_i would

be set of all visits other than itself. Though both decision variables encode the same problem, the two approaches vary because they try to exploit their respective solution strengths.

Constraints are at the core of all CP solvers. In any CP framework, constraints are *actively* used to reduce the domain of the variables. In simple terms, this amounts to removing values of variables which will not occur in any feasible solution. The values to be removed are decided using existing domains of variables and the constraints connecting them. This is known as *constraint propagation*. For example, consider the CSP defined by three variables, x,y and z, and binary constraints, C1, C2 and C3, represented in a constraint graph (Figure 2.3). The initial variable domains are specified in curly braces without a superscript. The constraints are propagated in lexicographic order, and the resulting domain, if different, is given in curly braces with the superscript of the constraint that was propagated.

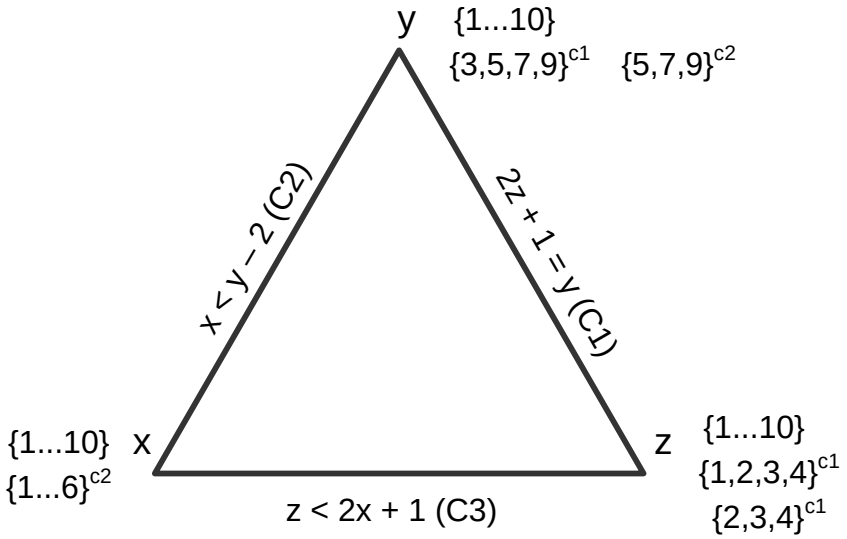


Figure 2.3: Constraint propagation

As shown in Figure 2.3, the domain of each variable is reduced by propagating the constraints. The interesting point is how constraints interact. After each constraint has been propagated once, we have the following domains: $x \in \{1...6\}$,

$y \in \{5, 7, 9\}$ and $z \in \{1, 2, 3, 4\}$. Now, when constraint C1 is propagated again the domain of z reduces to $\{2, 3, 4\}$. This happens because C2 removes 3 from the domain of y, in the first round of constraint propagation. This is a simple example of how constraints in CP interact via the domain of variables. It should also be noted that the order of propagation does not matter. Therefore, variable and domain declaration during the modelling phase is extremely important, as these factors dictate the effectiveness of constraint propagation.

The final domains for x,y and z are *arc consistent*. That is, for each binary constraint C_{ij} over variables (x_i, x_j), and for each value $v^1 \in D_i$, there exists a value $v^2 \in D_j$ which together satisfy the constraint. In general, this is termed *local consistency*. In the special case of binary constraint, as in our example, this is referred to as arc consistency. For constraints with higher arity than two, this is referred to as *hyper-arc consistency* or *Generalized Arc Consistency (GAC)*.

Note that for a constraint C_{ij}, whenever a value a in the domain of variable x_i does not have a corresponding value in the domain of x_j, the value a is removed from D_i as it is said to be *arc inconsistent*. If there did exist a value $b \in x_j$ which led to the feasible combination (a, b), then b would be referred to as the *support* of a.

Though achieving GAC leads to the maximum reduction in the domain of a variable, it can be time consuming. Since constraint propagation is interleaved with search, it is important to trade off the time spent on constraint propagation with time spent on search. For this reason, a weaker form of propagation called *bounds consistency* is widely used. Here only the bounds on the domain are reduced via propagation. This leads to improved performance in large problems such as VRPs.

Another, important concept in CP is that of *global constraints*. Global constraints are defined over a set of variables, and have specialized algorithms which

exploit, as the name suggests, the global structure of the constraints. This leads to more efficient constraint propagation than if the global constraint were implemented using smaller relations. For example consider the constraint over a set of variables $x_1, ...x_n$, which states that each variable is assigned a different value. This can be posted as ${}^nC_2$, constraints of the type $x_i \neq x_j$. The all-different global constraint in CP implements the same but more efficiently, and is written as $all - different(x_1, ...x_n)$. In addition to increased propagation it also promotes ease of modelling.

CP has also been used to solve optimization problems. The main drawback is that bounds on the objective function produced by constraint propagation are weak. Therefore, linear relaxations of global constraints are widely used to strengthen these bounds. This is also a popular method for integrating CP and OR. In the same vein, heuristic methods used for large scale problems, can benefit from constraint propagation. Search is performed by a local improvement method, while a CP framework is used to check feasibility. Constraint propagation is used to reduce variable domains, which in turn helps the local search. Since CP algorithms work at the constraint level, this leads to a robust heuristic design and is another avenue for integrated methods (Hooker [24]).

We use the latter method of integration to solve the collaborative vehicle routing problem in Section 3.9.

2.5 Local Search

This section introduces the concepts of local search as used in the thesis. Local search has been a well studied field, as a majority of real life problems cannot be solved using complete (exact) methods. Exact methods are those that guarantee an optimal solution when used to solve a problem. Understandably, local search

methods do not make this guarantee. However, this does not make these methods inferior, since high quality solutions are sufficient in most cases. This may also stem from the fact that, input data itself contains a certain amount of error. Therefore, when we claim optimality, it may not correspond to the actual problem. For this reason and many other practical considerations such as running time, local search is widely used and has also been used to solve the collaborative vehicle routing problem.

In the following, we present a general framework for local search (Van Hentenryck and Michel [23]).

2.5.1 Local Search Framework

An optimization problem $\wp$ can be defined by an objective function, f, and a set of constraints $C = C_1, , C_n$ over a set of variables $\vec{x}$

$$\wp = \{f(\vec{x}) \mid C_i : A_i x_i \leq b_i, 1 \leq i \leq n\}$$

A solution s to $\wp$ is an assignment of values to all variables in $\vec{x}$. A feasible solution to $\wp$ is a solution s that satisfies the conjunction of constraints $\bigwedge_{i=1}^{n} C_i$. The set of feasible solutions is given by $FS_\wp$. The set of optimal solutions to $\wp$ is defined by

$$OS_\wp = \{s \in FS_\wp \mid f(s) = \min_{k \in FS_\wp} f(k)\}$$

Local search algorithms used to solve $\wp$ have an objective function f to optimize. They start from an initial solution s and move to another solution in its neighborhood $N(s)$, subject to legality restrictions, and improvement in its objective function. This implies that a neighborhood need not be defined to include only

feasible solutions. In other words, the search space may be defined over a subset of C. To handle this, local search algorithms usually define a legal set $L(N(s), s)$, indicating the set of legal moves in the neighborhood of a solution s. The selection operator S chooses the next solution to move to as follows:

$$s = S(L(N(s), s), s)$$

This operator selects a solution from the legal set of solutions that can be reached from s. A local search algorithm stops either at a global optimal solution or more often at a local optimum. A local optimum is defined with respect to a neighborhood N. Given two neighborhoods N_1 and N_2, a solution s which is a local optimum in N_1, need not be a local optimum in N_2. This depends on the resulting transition graph of a neighborhood, and therefore neighborhood definition is extremely important.

Three neighborhood properties are usually taken into account during the design of a local search algorithm, namely neighborhood size, neighborhood connectivity and neighborhood constraints. The neighborhood size is usually a tradeoff between high-quality solutions and exploration time. Larger neighborhoods usually lead to better solutions, at the expense of greater computational time. The next property, neighborhood connectivity, has two types, weakly connected and optimally connected. A neighborhood is weakly connected if there exists a sequence of moves to reach an optimal solution s^* from any solution s in the search space. In contrast, an optimally connected neighborhood must have a path between any pair of solutions, s_1, s_2 in the search space. The neighborhood constraints focus on whether a search space should contain only feasible moves, or if infeasible moves should be allowed as well (i.e. if all or a subset of constraints in C should be considered).

The above description is concise and does not define local search rigorously.

However, the preceding definition highlights the flexibility available when designing a local search heuristic. In the next section, we describe classical heuristics which can be used to find an initial solution to vehicle routing problems.

2.6 Classical Heuristics

As the number of cities increases, exact solutions for the VRP become impossible except in a few cases with special structure. However, good feasible solutions to the VRP are usually sufficient. A well designed heuristic will in most cases lead to such solutions. Heuristics can be broadly classified into two categories: Classical Heuristics and Meta-heuristics. The former has the advantage of simple implementation and leads to good solutions in less computational time. Meta-heuristics are very much a research topic today, and are known to perform better than the classical heuristics, but at the expense of complicated implementation and greater run time. Metaheuristics which can be used to improve upon the initial solutions from classical heuristics are presented in the next section.

This section will focus on classical heuristics, for which there are many research articles. A good review of VRP heuristics is given in Laporte et al. [30]. The heuristics described below are used to find initial solutions for the VRPs we solve. The most popular classical heuristics are:

1. Clarke and Wright (Savings Heuristic)
2. The Sweep Algorithm
3. Petal Algorithms
4. Cluster First, Route Second Algorithms

The preceding heuristics can be mathematically described using the generic local search framework from section 2.5.1. Each of these Heuristics are briefly explained below.

2.6.1 Clarke and Wright Algorithm

The Clarke and Wright algorithm is a very effective heuristic when dealing with VRPs without time windows or with loose time windows. Suppose we have two nodes i and j which we had to visit. If we travelled to each node separately, the total distance would be $(d_{0i} + d_{i0} + d_{0j} + d_{j0})$. In contrast, if we visit i and j on the same route, the distance for this route will be $(d_{0i} + d_{ij} + d_{j0})$. Therefore, the savings, defined as the improvement due to visiting i and j together on the same route, is

$$S_{ij} = (d_{0i} + d_{i0} + d_{0j} + d_{j0}) - (d_{0i} + d_{ij} + d_{j0}) = d_{i0} + d_{0j} - d_{ij}$$

The Clarke and Wright algorithm combines routes greedily based on savings as follows:

Step 1: Compute the savings for each pair and order this in a non-increasing array. Create n vehicle routes $(0, i, 0)$ for $i = 1$ to n, with a vehicle of capacity C_0 serving each route.

Step 2: Starting from the top of the savings list, execute the following. Given a saving S_{ij}, determine whether there exist two routes, one starting with $(0, j)$, and the other one ending with $(i, 0)$, that can be merged feasibly. If so, combine these two routes by deleting $(0, j)$ and $(i, 0)$ and introducing (i, j).

2.6.2 Sweep Algorithm

Assume each vertex is represented by its polar co-ordinates (θ_i, ρ_i), where θ_i is the angle and ρ_i is the ray length. Assign a value $\theta_i^* = 0$ to an arbitrary vertex i^* and compute the remaining angles centered at 0 from the initial ray $(0, \rho_i^*)$. Rank the vertices in increasing order of their θ_{i^*}.

1. (Route initialization). Choose an unused vehicle k.
2. (Route construction). Starting from the un-routed vertex having the smallest angle, assign vertices to vehicle k as long as its capacity or the maximal route length is not exceeded. If un-routed vertices remain, go to Step 1.
3. (Route optimization). Optimize each vehicle route separately by solving the corresponding TSP (exactly or approximately).

2.6.3 Petal Algorithm

This is a variation of the sweep algorithm. A number of routes, referred to as petals, are created in a similar manner to the sweep algorithm. The two methods differ in the route optimization stage. The petal algorithm uses a set partitioning algorithm.

2.6.4 Cluster First, Route Second Algorithms

Instead of using a geometric method to form the clusters, this method solves a Generalized Assignment Problem (GAP). Route construction can be performed by solving a TSP within each cluster.

2.7 Metaheuristics

A complete survey of meta-heuristics applied to VRP and VRPTW can be found in Bräysy and Gendreau [6, 8]. This section will give a brief introduction to tabu search (Glover [18]) and guided local search (Voudouris and Tsang [47, 48]), which are the two metaheuristics used in this thesis. Following the notation of section 2.5.1, let $S_{\wp}$ denote the set of problem solutions. The basic local search template given in Table 2.1 will be used for the purpose of illustration.

Choose initial solution s in $S_{\wp}$
$s^* = s$
While not STOP **do**
$k = k + 1$
Choose S^* solutions in $N(s, k)$
Find best $s^{'}$ in S^*
if$(f(s^{'}) < f(s))$ $s^* = s'$
End While

Table 2.1: Basic local search template

The search starts with an initial solution. It then moves to a best solution in the neighborhood of the current one, provided the new solution has a better objective. A stopping criteria STOP is used to terminate the search.

Metaheuristics are widely used in combinatorial optimization problems to obtain high quality solutions in reasonable time. Each metaheuristic has its own unique way of using information from neighborhoods and solutions visited in the past, to avoid locally optimal solutions and search the solution space efficiently. In this vein, almost all metaheuristics have a method to diversify the search to unvisited regions, or intensify the search in promising areas of the solution space. The following subsections deal with tabu search and guided local search.

2.7.1 Tabu Search

Tabu search (TS) has its origins in artificial intelligence and was proposed at the same time in both Glover [18] and Hansen [22]. Tabu search is a metaheuristic which has been successfully used to solve various problems in combinatorial optimization. In particular, it has been successful in solving VRPTWs.

A local search method moves from a solution s to another solution $s^{'}$ in the neighborhood $N(s)$ of s. A simple iterative improvement scheme, such as a descent method, would choose the best solution $s^{'}$ in $N(s)$ at every iteration and move to it. When choosing the solution $s^{'}$, we need to decide if the entire neighborhood will be searched, or if only a subset S^* of solutions in $N(s)$ will be scanned. As an example, if $S^* = N(s)$, we scan the entire neighborhood, which would be prohibitively time consuming for large problems. On the other extreme, we can set $|S^*| = 1$, in which case the best element will always be an arbitrary element in the neighborhood.

Therefore, efficient heuristics search for solutions in the neighborhood of an existing solution in a strategic manner. Tabu search maintains a *tabu list*, based on selected attributes of moves and solutions, which controls the solutions it scans in $N(s)$ at every iteration. This is equivalent to using information from previous solutions and neighborhoods, making it an informed search method. A simple tabu list could be based on "recency", where moves which were recently visited are forbidden. Usually, multiple tabu lists are maintained, and multiple criteria are evaluated to decide if a move is "tabu" (i.e. is forbidden)). For more detail, the reader is referred to Glover [19, 20].

A tabu list will certainly help direct the search at every iteration, but the following two situations will still be encountered and must be resolved.

1. A tabu list that is too large, in addition to requiring extra memory overhead, will also lead to a highly restricted search. On the other hand, if the list is

too small, it will have little or no effect in directing the search. In both cases, the purpose of having a tabu list will be defeated.

2. Regions of a solution neighborhood which are tabu may actually contain the best solution. This may lead to the best solution never being found, or being found much later in the search.

Situations 1 and 2 can be dealt with through two important features of the tabu list, *tabu tenure* and *aspiration* which will now be explained. Tabu tenure is the length of time, usually measured in number of iterations of the heuristic, for which an item in the list remains tabu. This is a parameter which needs to be tuned when using a tabu search, and may be used to address the first concern above. Aspiration levels are set to accept highly attractive solutions which are tabu. A commonly used aspiration criteria accepts any tabu solution which is better than the best solution found so far. This solves the second problem mentioned above.

In general, intensification and diversification schemes (Rochat and Taillard [36]) have been developed to address the first issue. Dynamically changing the length of the list or tenure during search is one such option, and this would make the tabu search reactive.

For the VRPTW, tabu search has performed extremely well; the implementations by Gehring and Homberger [17] and by Cordeau et al. [10] are among the best. The former paper uses a parallelized tabu search. It employs an evolutionary algorithm which utilizes Or-opt, $2-opt^*$ and λ-interchange moves, while a specialized Or-opt is used for reduction of vehicles. The initial solution is found via a stochastic variant of the Clarke and Wright algorithm (refer to Section 2.6.1). Similar to other metaheuristics, the primary objective is to reduce the number of vehicles, while the secondary objective is to minimize the distance.

The tabu search in [10] is simple and effective. Cordeau's implementation allows

infeasible solutions at a penalty. A simple insertion operator, which relocates a customer, is used for local search; a variant of the sweep algorithm (Section 2.6.2) is employed to instantiate the routes. The objective is the same as in [17]. Post optimization is performed through a heuristic for the travelling salesman problem. On the problem set from Solomon [41], the results of both [17] and [10] were only 2-3% percent worse than the best known solutions.

The heuristic in De Backer and Furnon [3] and De Backer at al. [4] requires a special mention, as a variant of it is applied in the present work. They use both a tabu search heuristic and a guided local search to solve VRPTWs (The latter method will be explained in Section 2.7.2). The neighborhood is defined by two intra-route operators, 2-opt and Or-opt, and three inter-route operators, cross, exchange and relocate. Two tabu lists are maintained, one for edges added and the other for edges removed. A constraint programming framework is used for checking feasibility. The search has a single objective, to minimize total travel distance. Understandably, the tabu search performs worst in terms of number of vehicles, but surprisingly, it only performs better than the other approaches in R2 and RC2 of the Solomon's benchmark. We attempt to enhance the implementation in [3].

Tan et al. [44] develop a tabu search heuristic and a simulated annealing heuristic for the VRPTW. They use a modified version of Solomon's insertion heuristic to find an initial solution. Local minima are avoided through a diversification scheme, which uses λ-interchange with a $2 - opt^*$ operator. Throughout the search, elite solutions are recorded, to be used as a starting point for intensification. References [3] and [44] furnish the only tabu search heuristics that minimize route distance, and will used for the performance evaluation of the proposed tabu search in Section 4.4.

2.7.2 Guided Local Search

Guided local search (GLS) (Voudouris and Tsang [47, 48]) is a rather recent addition as a metaheuristic. In GLS, the objective function is modified to guide the search. Once again, the search requires memory to remember and use the information it gains. For this purpose, a set F, of features of a problem is defined. An indicator function $I_i(s)$ and a penalty factor p_i, are defined for each feature. For each candidate solution, $I_i(s) = 1$ if feature $i \in F$ is present in solution s, and 0 otherwise. The penalty p_i is initialized to zero, and keeps track of how many times feature i has occurred so far. With the above definitions and a penalty factor, λ, which is the only parameter of the search, the modified objective function can be written as

$$h(s) = f(s) + \lambda \sum_i p_i I_i(s) \tag{2.1}$$

In equation 2.1 each feature has a cost penalty of one. However, we may want to penalize each feature differently using a cost vector c, where c_i is the cost of feature $i \in F$. Penalties only serve to diversify the search and not for intensification. The most interesting aspect of GLS is that it penalizes only a subset of the features using a novel feature selection mechanism. A utility value $U_i(s)$ is calculated for each feature, where

$$U_i(s) = I_i(s) \frac{c_i}{1 + p_i} \tag{2.2}$$

The above equation will result in higher values of $U_i(s)$ for those features which exist in solution s but have not been previously penalized. Each time a feature is penalized, the corresponding p_i is incremented by one. However, if a feature has been encountered often, it is penalized less. In other words, the equation promotes candidate solutions with "good features," where a good feature is one which occurs often at a local optimum, and therefore has a high penalty value, p_i.

GLS has the advantage of having to configure only the single parameter, λ.

Therefore, tuning a GLS heuristic is much easier than tuning a tabu search, which is far more complex. Similarly, one can decide much faster if GLS is suitable for a given problem much faster than one can for tabu search; this is an added bonus. De Backer et al. [4] use GLS as a diversification scheme on top of tabu search and report excellent results on the benchmark problems of Solomon [41], Taillard et al.[43] and Fisher. Three new best solutions are reported for Solomon's data set. They conclude that guided local search, together with the tabu search, gives the best performance on the benchmark problems that they solved.

A drawback of their implementation is that the number of vehicles is reduced using knowledge from existing best known solutions. They begin with as many vehicles as there are in the best known solution. Dummy vehicles with high cost are then added to ensure feasibility. This discourages solutions whose number of vehicles exceeds that in the best known solution. More importantly, the preceding logic cannot be applied to real life problems, such as the those solved in this thesis, for which no advance knowledge is available of the best solutions. Results reported using such a method do not reflect the "real" performance of the heuristic, hence we refrain from using their approach for evaluation purposes.

Chapter 3

Problem and Proposed Solution

The necessary background for the thesis was established in the last chapter. This chapter describes in detail our proposal for carrier collaboration in section 3.1. Examples of simple cases where collaboration can be beneficial are also given. Once the framework is complete, the collaborative vehicle routing problem is mathematically defined in section 3.2. Section 3.3 presents some geometric results for a restricted version of the COL-VRPTW. Interesting reasons are given for carriers to collaborate in 3.4. Sections 3.5- 3.8 present heuristics related to the location of transshipment facilities and collaborative routing respectively. Finally, a summary of the overall solution procedure in given in Section 3.9.

3.1 Carrier Collaboration

The linehaul truckloads that arrive at a breakbulk point are disaggregated into smaller shipments and delivered using LTL carriers. As LTL carriers performing the local deliveries serve geographical regions that are compact, especially in urban freight transportation, this leads to overlapping routes.

If routes overlap, and vehicles are travelling less than full, there may exist

opportunities for carriers to collaborate, improve asset utilization and reduce asset-repositioning costs. This question led to the investigation of carrier collaboration. The benefits to LTL carriers of carrier collaboration are similar to the benefits to TL carriers of *shipper* collaboration. However, the underlying process is quite different, and there is a need to study carrier collaboration independently. It should also be noted that not all shippers will be able to tender TL loads and benefit from TL collaboration. Therefore, LTL collaboration may be seen as having wider applicability, though assembling these collaborative routes is more complex than in the TL case.

Carrier collaboration initiatives are important, since pressure to ship sooner (i.e. with shorter lead times) has led to partially loaded trucks delivering goods in urban regions. This poor asset utilization leads not only to low carrier revenues, but also to greater congestion in cities. Congestion in urban areas is becoming an increasingly important problem for both commuters and environmentalists.

Urban areas are thus particularly suited for carrier collaboration. In many cities, there are a limited number of points of entry for trucks. (In Toronto, for example, there are about four points of entry from major highways.) Breakbulk points often are located at these entry points. Once carriers break down their loads into smaller shipments there, they may combine some of those loads, resulting in fewer trucks entering the city. This would also lower congestion. Feasibility of such operations depends on spatial orientation of routes, time windows, and truck capacity, which all result in a complex problem.

Before proceeding further we define two important terms. *Deadhead miles* are those that the truck travels empty, while *extra miles* are those that the truck travels inefficiently. Consider carriers A and B. Carrier A is travelling extra miles if carrier B can accommodate a customer of carrier A on its route, and incur a smaller increase in its route distance than the decrease in route distance of carrier A. This

is the inefficiency indicated in the definition of extra miles. Therefore, reducing extra-miles can lead to a decrease in pollution.

Next we explain the details of carrier collaboration. For simplicity we assume no time windows in the examples to follow. One of the possibilities for collaboration occurs when the routes of each carrier overlap (Figure 3.1), and one of the trucks (red/solid route) has sufficient space to accommodate the loads of all customers from another carrier's route (black/dashed route). In this case, the black carrier saves a truck, and has lower deadhead and extra miles, while the red carrier has higher asset utilization and lower deadhead miles. The decrease in cost can be shared between the collaborating carriers.

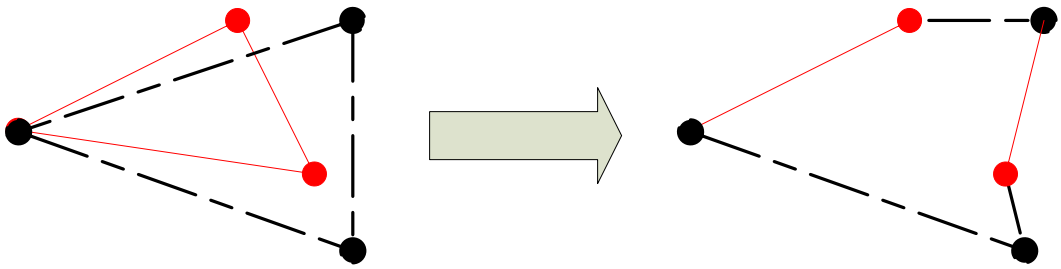


Figure 3.1: Collaboration results in the reduction of vehicles, extra miles and empty miles, while increasing asset utilization

Due to lack of space, it may not be possible to accommodate all the customers from another carrier's route. However, assuming that some customers can be accommodated, collaboration may still be beneficial if it can result in reduction of extra miles travelled.

To illustrate, consider the two routes shown in Figure 3.2. The nodes of the red/solid route marked R1, R2 and R3 are the customers that can be profitably transferred to the black/dashed route. After collaboration, the resulting routes (shown on the right) allow the red carrier to reduce extra miles travelled, and allow the black carrier to reduce deadhead miles while increasing its asset utilization. If revenues can be shared, this exchange is mutually beneficial.

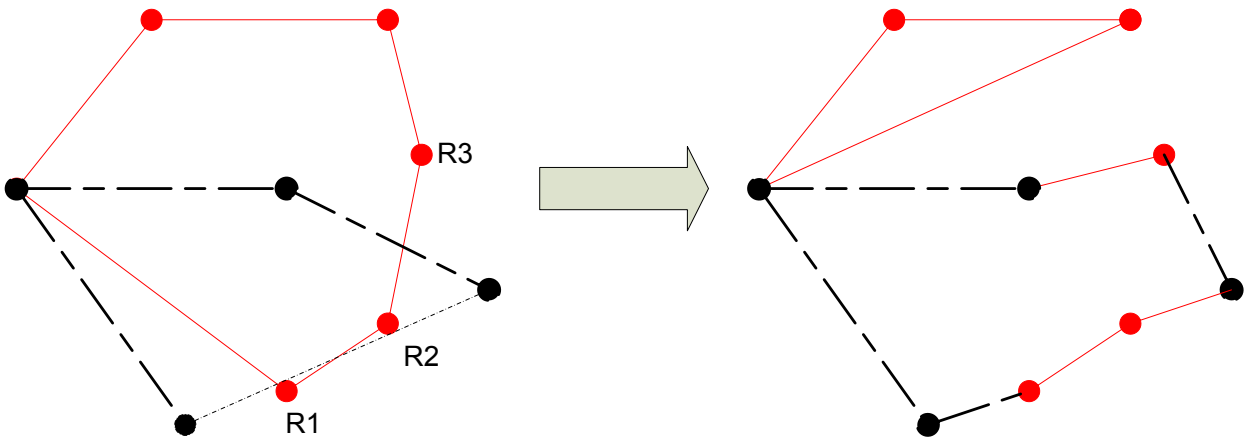


Figure 3.2: Collaboration reduces extra miles and deadhead miles, while increasing asset utilization

The situations described above, of collaborating and transferring loads at an entry point, may not be possible if trucks are full and have large loads to be delivered early in the route. Once these initial loads are delivered, the trucks will have excessive empty space and low asset utilization. To counter this disadvantage we need to make the collaboration more dynamic. Carriers must be able to transfer goods, if possible, after they have left the depot and while they are still performing deliveries.

A stylized example of this is shown in Figure 3.3. The transshipment point is marked TP. B1 is the node on the black/dashed route from which the truck deviates to pick up the loads of R1 and R2 at node TP. The tradeoffs here are the increase in deadhead miles as a result of the deviation to pick up loads at TP, and the reduction in extra miles as a result of collaboration. The new routes resulting from collaboration do not overlap. Again, collaboration leads to a win-win situation for both carriers. Figure 3.3 considers transshipment at a customers site, but the transshipment point need not coincide with a customer.

Once time windows are introduced, collaboration results in yet another advantage. By transferring customers from one route to another, we will definitely reduce the routing time of the former carrier. Although this may increase the routing time

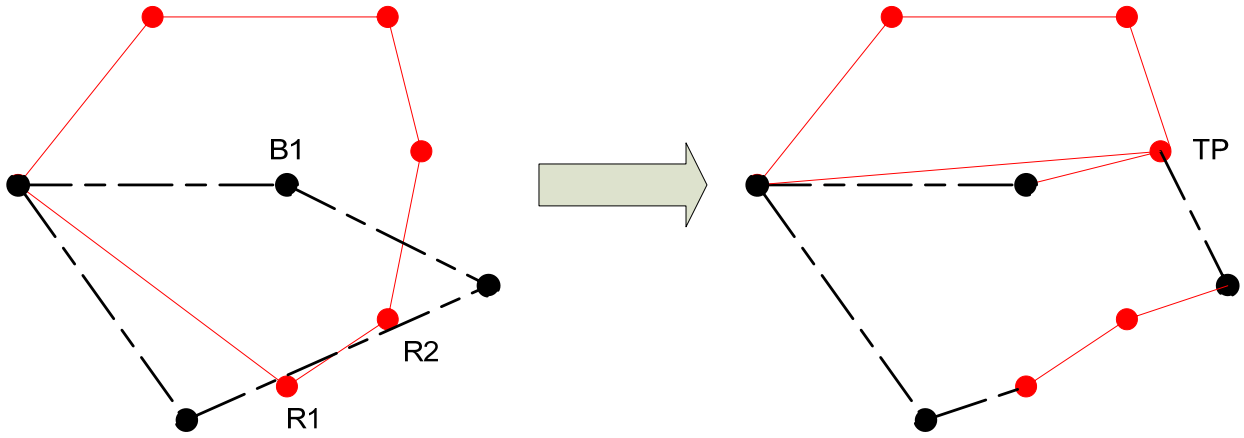


Figure 3.3: Transshipment at the customer's site

of the other, the route from which customers are transferred will definitely have a shorter routing time. In addition to fewer extra miles, the associated truck will be available for reuse much earlier. This may allow the same vehicle to be employed for the next set of local deliveries. Therefore, we not only save in distance but also in time, which may translate to better service and higher overall asset utilization. In fact, even the carrier to which passengers are transferred will enjoy increased asset utilization. The case of time windows will be handled in detail when describing the local search algorithm in Section 3.8.

The features and benefits of carrier collaboration have been defined using simple examples in this section. However, the implementation of collaboration is quite challenging, and requires strong commitment from all carriers in the community. Further, efficient transfer of goods is also crucial in achieving substantial gains through collaboration.

3.2 The Collaborative Vehicle Routing Problem

Vehicle routing problems (VRP) or their mathematical equivalents arise in every day life. Solutions of these models improve or enable the use of a telephone, travel

for business, receipt of mail, etc and have been extensively studied. The reader is referred to Laporte [29] for an introduction and to Toth and Vigo [46] for an in-depth review of the VRP. As mentioned in the previous section, LTL collaboration during routing can be modeled as a variant of the vehicle routing problem. This will be referred to as the COLlaborative Vehicle Routing Problem with Time Windows (COL-VRPTW).

COL-VRPTW has some features of the Multi Depot Vehicle Problem (MD-VRP), the Vehicle Routing Problem with Time Windows (VRPTW), and the Pickup and Delivery Problem with Time Windows and Transshipment (PDPTWT). The capacitated vehicle routing problem (CVRP) is one of the simplest variants of the VRP. CVRP concerns a fixed fleet of delivery vehicles of uniform capacity which must service known customer demands for a single commodity from a common depot at minimum transit cost. The VRPTW is similar to the ordinary VRP, with the additional constraint that each customer should be supplied within a specified time window. In the PDPTWT, each customer has pickup and delivery locations associated with it, and goods may be transshipped at pre-specified points.

Mitrović-Minić and Laporte [34] proposed an insertion heuristic for the PDPTWT. They develop a two-phase heuristic consisting of a construction phase followed by an improvement phase. The paper concludes that the advantage of transshipment increases with problem size and with time window size. However, they reported that for random instances, the transshipping of goods is not very beneficial.

A partial reason for this non-satisfactory result is the fixing of transshipment points a priori, without any consideration of the structure of the particular instance. It should also be noted that the transshipment itself is quite different in their case and in ours. In the PDPTWT, transshipment takes place by splitting a pickup and delivery request. A truck first drops off the load at a transshipment point. Following this, another truck handles the request from the transshipment point to

the destination.

In the COL-VRPTW, the requests are not of the nature of a pickup and delivery pair. Goods are loaded at the depot and are destined to be delivered to customers, with the allowance of exchanging goods with other carriers at transshipment points. Further the transshipment is between two trucks belonging to different carriers. This restriction is added since we are interested in the benefits from inter-carrier collaboration. It can be relaxed otherwise.

For these reasons the, COL-VRPTW has distinct features which cannot be handled by the solution methodologies designed for the aforementioned problems. COL-VRPTW arises in urban cities where the routes of different carriers overlap, and the aim is to exploit goods transfer between collaborating carriers in a mutually beneficial manner. Different scenarios for this have been explained in the previous section through simple examples; additional possibilities specific to the time window case will be shown in a later section.

To the best of our knowledge, the COL-VRPTW has not been studied until now, and therefore no literature on solving the same problem exists. We suggest a formal definition of the COL-VRPTW. In the following definition we assume that there are p points of entry into the city. For example, p would be equal to four, in a city like Toronto. In VRP terminology, this would correspond to p depots. They may be taken as the break-bulk points discussed previously. Further, these depots are added as the last p nodes in the customer set. The required sets and problem definition follow:

Definition 3.2.1. COL-VRPTW: Let us assume a set of K carriers, $C = c_1, \cdots, c_K$; a set of N_c customers for each carrier, $I_c = \{i_1^c, \cdots, i_{N_c}^c\}, c \in C$; a set of NT transshipment points $T = \{t_1, \cdots, t_{NT}\}$, and a set of NV_c vehicles of equal capacity, $V_c = \{v_1^c, \cdots, v_{NV_c}^c\}$, $c \in C$. The carriers' depots are given by the,

$D = \{D_1, \cdots, D_K\}$, where $D_i = \sum_{c=1}^{K} N_c + i$. Duplication of depots for each carrier, allows its set of vehicles to have unique starting and return times at the depot. The set $UC = [\sum_{c=1}^{K} I_c] \bigcup D \bigcup T$ is the universal set containing all customers, depots and transshipment points. Each customer $j \in UC \setminus T$ has an associated time window $[E_j, L_j]$, where E_j is the earliest time at which service can start and L_j is the latest time at which service can begin. The earliest and latest start times E_d and L_d, $\forall d \in D$, correspond to the earliest time at which the carriers' trucks can leave the depot, and the latest time by which they must return to the depot respectively. Along with the nodes (customers) of the problem, there is a set A of arcs with non-negative weights (distances) and associated travel time t_{ij}.

A vehicle is allowed to arrive before E_j and wait at no cost until service becomes possible. The time $b_j, j \in \bigcup_{c=1}^{K} I_c$, at which service begins at a customer is a decision variable. Each node $j \in [UC \setminus \{D \bigcup T\}]$ imposes a service requirement, q_j, that is a delivery or a pickup. A route, r_{c_1}, of carrier $c_1 \in C$ is allowed to serve customers in the route r_{c_2} of a carrier $c_2 \in C$ (i.e. a customer in the set $\{j | j \in I_{c_2}\}$), if and only if r_{c_1} meets r_{c_2} at a transshipment point $t \in T$ and is able to collect the load corresponding to the customers being transferred from r_{c_2} to r_{c_1}. The objective is to find the minimum cost set of tours R^* for a set of identical vehicles, such that all nodes $j \in UC$ are served within their time windows, and the accumulated load up to any node L_j does not exceed a positive number Q, the vehicle's weight capacity.

We assume that all distances and travel times satisfy the triangle inequality. The conventional way of handling a large VRP instance has been through two-phase heuristics. The first phase focuses on route construction, while the second phase is a post-optimization phase, which usually involves a variant of 2 and 3 arc-exchange moves. Local search methods have been applied to a variety of VRPTWs with great success (Mester and Bräysy [33], Rochat and Taillard [36], Taillard et al. [43]). The papers by Cordeau et al. [10] and Bräysy and Gendreau [7, 8] provide

an excellent review of heuristics and exact algorithms for the VRPTW.

Effort has also been guided towards using constraint programming techniques, namely constraint propagation with local search to improve the efficiency of the local search scheme (Caseau and Laburthe [9], De Backer et al. [4], Kilby et al. [25]). The basic idea is to perform constraint propagation and local optimization of routes during tour construction itself (e.g. in the case of an insertion heuristic for route construction, tour optimization and constraint propagation are performed after every insertion). Experiments in [9] with such integration establish that it provides substantial improvements in performance and quality for large vehicle routing problems.

The COL-VRPTW does result in large problems due to the involvement of multiple carriers. Further, because of the interaction between multiple-carrier routes at transshipment points, it involves side constraints which constraint propagation can handle efficiently. The side constraints include:

1. **Inter-carrier Constraints:** Only customer loads from routes of different carriers can be transferred at transshipment points, when they enter into a collaborative exchange. This restriction is imposed to allow the evaluation of the benefits from collaborative exchanges between different carriers alone. If this requirement were relaxed, the resulting version would be unrestricted in allowing both traditional transshipment between routes of the same carrier and collaborative exchanges between carriers.

2. **Precedence constraints:** The truck of the carrier giving the load ("Load Giver") must arrive at the transshipment point before the truck of the carrier taking the load ("Load Taker"). In our implementation, we restrict the difference in arrival times of the two carriers, to avoid storage of inventory at the transshipment facility.

3. **Load constraints:** The inter-route and inter-carrier transfers at transshipment points must result in routes which satisfy the weight capacities of the collaborating vehicles.

4. **Savings constraints:** For a transfer to be successful, the resulting savings should be positive. Savings need not necessarily imply a reduction in routing distance. A transfer resulting in negative distance savings may lead to a large reduction in the load giver's routing time. Reduction in total route time will give the load giver added potential to take part in good collaborative exchanges in later iterations of the algorithm. This may lead to an overall positive distance savings, or a savings in the number of vehicles as the vehicle corresponding to the load giver will be able to return to its depot at an earlier time and perform another set of local deliveries.

   The above explanation takes the liberty of assuming that the load giver's partial route will be feasible when one or more customers are removed. A sufficient condition for this is that the travel times satisfy the triangle inequality.

The important difference between local search and *constrained* local search is the active use of constraints (constraint propagation) to reduce the search space. As an alternative, the moves of a local search algorithm can be modified to check for the complicated infeasibilities introduced by the side constraints. However, it is more beneficial to remove invalid moves through propagation, than to complicate the move itself. The former strategy can lead to savings in the running time of the algorithm.

In our methods, we solve the problem of carrier collaboration during local delivery in a city where carrier routes overlap. The opportunities for carrier collaboration are not limited to this situation alone. A mirror problem related to the pick-up of loads can be solved independently using the heuristics we propose. However, in

that case the time windows will be those for *dispatch*.

We either use a modified tabu search algorithm or a guided local search algorithm to solve for the routes specific to a carrier's VRPTW. As stated above, many excellent local search methods have been proposed for the VRPTW, and we do not attempt to design a heuristic for this purpose. Our effort has been focused on finding good collaborative routes. However, our work is similar to the preceding references in using constraint programming and optimization techniques with in heuristics, wherever such integration proves beneficial in reducing the search space or in improving solution quality. If the initial route building for each carrier were to be considered, our proposed method would become a three-phase heuristic.

3.3 Geometric Results

Collaborative vehicle routing involves solving large vehicle routing problems with hundreds, or even thousands, of customers. When the number of customers is this large, it is extremely difficult, if not impossible, to derive analytical results. Therefore, we analytically study simple cases, the results of which can be used to characterize the savings from larger problems solved using heuristics, or to design the heuristic itself. Route overlap is defined as

Definition 3.3.1. Route Overlap: An overlap of routes between carriers C_A and C_B implies that some customers of C_A lie within the boundaries which define one of the routes of C_B, vice versa or both.

The term "collaboration" is used to characterize a symbiotic relationship between carriers. Consider two carriers C_A and C_B. Collaboration implies that C_A serves some customers of C_B or vice versa, or both. Both collaborating partners benefit because savings are assumed to be shared. In this section, we present inter-

esting geometric proofs of savings for simple collaborative scenarios. The central idea behind the proofs is given in Remark 3.3.1, the proof of which is simple, and has hence been omitted.

Remark 3.3.1. *Suppose we have shown that savings are positive for all points within a region for a particular route r. Although, there may or may not be another route with savings greater than that of route r, there exists at least the latter route for which the savings is positive at all points within the region. In other words, collaboration will be beneficial within this region.*

Theorem 3.3.2. *For two carriers having a common depot located at a corner of a square, and one of the carriers having customers located at any two of the three remaining corners of the square, a positive savings will result due to collaboration as long as the other carrier's customer is located within the square as well.*

Proof. Let the two carriers be C_A and C_B. C_A has two customers A_1 and A_2, while C_B has just one customer B_1 (Figure 3.4). The common depot of C_A and C_B is represented by O. The initial routes are $O - A_1 - A_2 - O$ and $O - B_1 - O$.

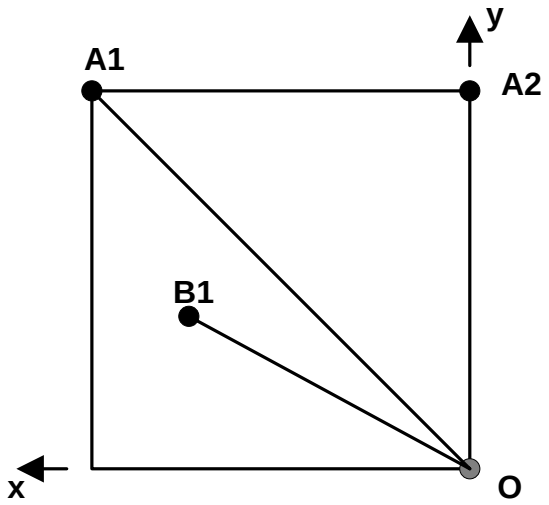


Figure 3.4: Collaboration between two carriers with the same depot

For the proof, we consider the collaborative route $O - A_1 - B_1 - A_2 - O$. The links added are $B_1 - A_1$, and the links severed are $B_1 - O$, $O - A_1$. This implies

that the savings from collaboration is

$$Savings = d(B_1 - O) + d(O - A_1) - d(B_1 - A_1) \tag{3.1}$$

In Eq 3.1, $d(a)$ returns the length of arc a. The savings is positive for any position of B_1, in light of the *triangle inequality*. Using Remark 3.3.1 and the symmetry of the square, the proof is complete. □

Conjecture 3.3.3. *Consider two carriers C_A and C_B. C_A has its depot O_A at the corner of a square, and has two customers A_1 and A_2 located at two non-diagonal corners of the three remaining corners of the square. A positive savings will result if carrier C_B has its depot on the edge connecting O_A and the uncovered corner, and if the routes overlap (Definition 3.3.1).*

Now we explain the reasoning behind Conjecture 3.3.3. Carrier C_A has two customers A_1 and A_2, while C_B has just one customer B_1. The respective depots are represented by O_A and O_B. O_A is located at a corner of the square, while O_B lies on the edge connecting O_A and an uncovered corner, as shown in Figure 3.5 (A unit square simplifies the analysis and the results for a general square can be obtained by scaling the results we present).

The customers are distributed such that A_1 and A_2 are on two vertices of the square, while B_1 overlaps with the route of C_A.

Consider the area represented by the triangle $O_1 - A_1 - A_2$. Let us assume that the coordinates of B_1 are (x_{B_1}, y_{B_1}), with the origin of the coordinate system located at O_A. The distance between O_A and O_B equals b, where $0 \leq b \leq 1$. The route $O_A - O_B - A_1 - A_2 - B_1 - O_A$ is the collaborative route for the analysis to follow. The initial or non-collaborative routes of C_A and C_B are $O_A - A_1 - A_2 - O_A$ and $O_B - B_1 - O_B$, respectively. The savings will depend on the distance of the

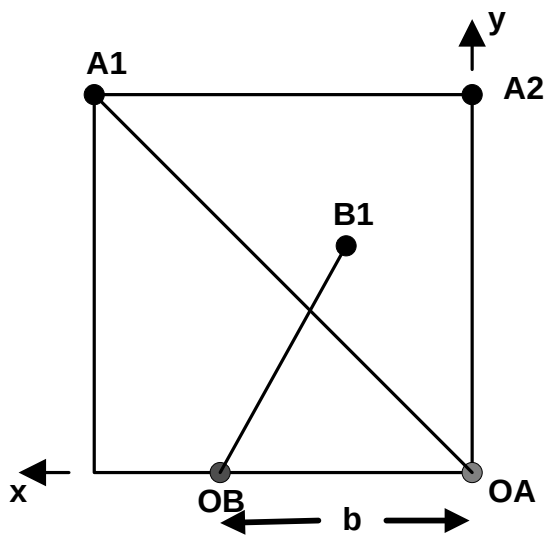


Figure 3.5: Collaboration between two carriers with unique depots

links added and removed.

$$Edges\ Added : O_A - O_B, O_B - A_1, A_2 - B_1, B_1 - O_A$$

$$Edges\ Removed : O_B - B_1, B_1 - O_B, O_A - A_1, A_2 - O_A$$

Let d(edges added) and d(edges removed) represent the distances of the sum of the edges added and removed respectively.

$$d(edges\ added) = b + \sqrt{(1-b)^2 + 1} + \sqrt{x_{B_1}^2 + (1 - y_{B_1})^2} + \sqrt{x_{B_1}^2 + y_{B_1}^2}$$

$$d(edges\ removed) = 2\sqrt{(x_{B_1} - b)^2 + y_{B_1}^2} + \sqrt{2} + 1$$

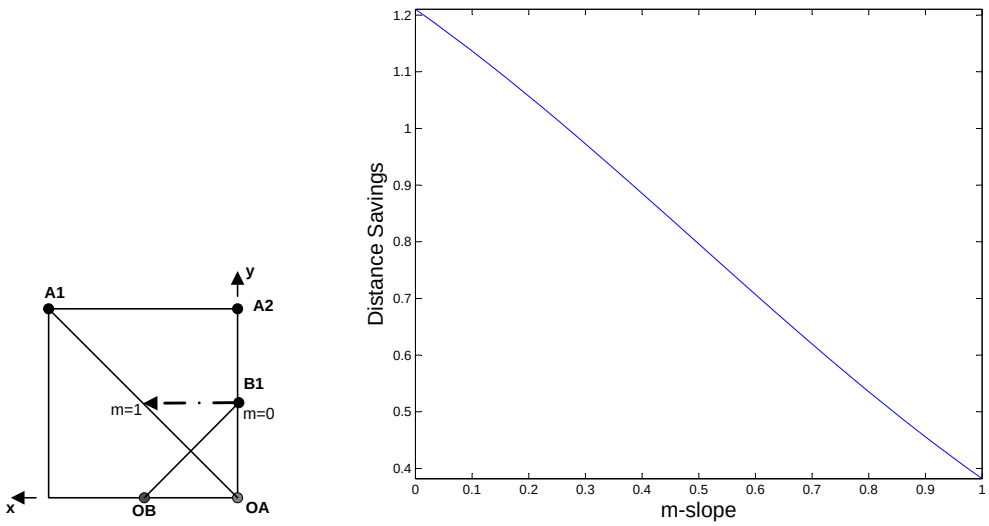
$$Savings = d(edges\ removed) - d(edges\ added)$$

At present, we are able to only graphically show that the above savings expression is positive for all values of $0 \leq x, y, b \leq 1$, which from Remark 3.3.1 implies that savings are positive when B_1 is located within the route of C_A. In other words, savings from collaboration are positive when the routes overlap. We are currently, trying to find either a transformation or a bounding function which will allow us

to analytically prove Conjecture 3.3.3.

3.3.1 Graphical Analysis

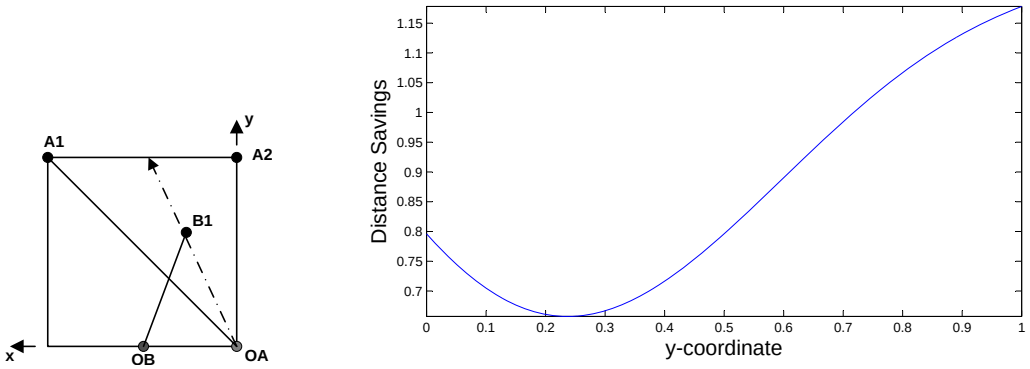
In this section, variation in distance savings will be analyzed by changing the parameters of the savings expression given by Eq **??**. Figure 3.6 is a plot of the variation of distance savings with change in m, after fixing $b = 0.5$ and $y_{B_1} = 0.5$. Recall that m is the slope of a ray starting from O_A and ending at the edge $A_1 - A_2$. Therefore varying m from 0 to 1 is equivalent to horizontally translating B_1 from $O_A - A_2$ to $O_A - A_1$, as shown in Figure 3.6 (a). The savings is a maximum when $m = 0$ and reaches its minimum value at $m = 1$. The collaborative route which we considered when deriving the savings expression is $O_A - O_B - A_1 - A_2 - B_1 - O_A$. Since we add B_1 in between A_2 and O_A, as m increases B_1 moves further away from this edge which leads to the reduction in savings. Therefore, during local search, it will be sufficient to consider only those customers that are close to an edge for insertion. It is for this reason that VRP insertion heuristics employ a "radius" parameter.



(a) **(b)**

Figure 3.6: Distance savings as a function of slope m, when $b = 0.5$ and $y_{B_1} = 0.5$

Next, we analyze the variation of distance savings with change in y_{B_1}. Again, $b = 0.5$, and we have fixed $m = 0.5$. This is equivalent to moving B_1 along the line with slope 0.5, starting at O_A and ending at the edge $A_1 - A_2$. As shown in Figure 3.7 (b) the savings initially decreases to zero and then starts to increase again, reaching a maximum at $y_{B_1} = 1$. This is due to the tradeoff between the distance saved by not travelling from O_B to B_1 and back, and the distance incurred by adding B_1 in between A_2 and O_A. If the same graph were plotted for $b = 1$, the savings would be a maximum at $y_{b_1} = 0$ and continue to decrease as y_{b_1} increases. Therefore, even though we select customers that are close to the edge for insertion, the relative position of a customer to its depot and to the customer defining the edge are also important in defining the savings.

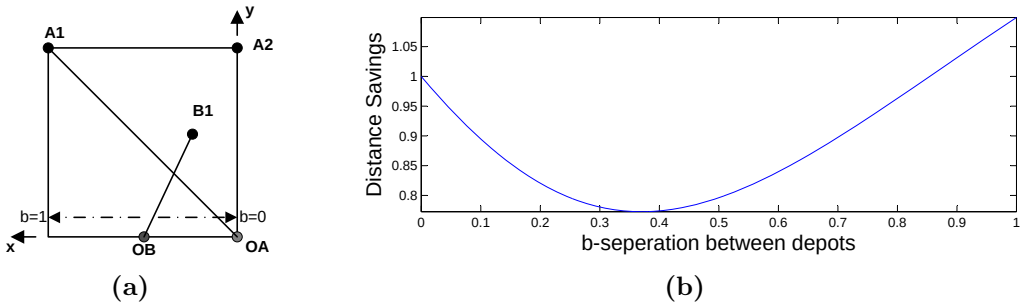


(a) **(b)**

Figure 3.7: Distance savings as a function of the y coordinate of B_1 (y_{B_1}), when $b = 0.5$ and $m = 0.5$

As shown in Figure 3.8, varying the distance b between depots, has a similar effect to varying y, due to the tradeoff between distance added and removed. The preceding results show that collaboration even in such a simple example, can be non-intuitive. That will certainly be the case for instances with less structure and more customers. However, simple geometric results, such as those just discussed, can be used to define a "relatedness" parameter, to restrict the number of customers considered for collaboration during local search. This will be explained in Section

3.8. In the next section, we highlight interesting benefits which result from carrier collaboration.



(a) (b)

Figure 3.8: Distance savings as a function of the distance between depots, when $y_{B_1} = 0.5$ and $m = 0.5$

3.4 Why Collaborate?

Before proceeding to explain the algorithms and solution procedures, we highlight some interesting reasons for carriers to collaborate within the structure of our proposed framework for collaboration (Section 3.1).

To better understand collaboration, we first explain the non-collaborative scenario. Trucks belonging to different carriers come to the entry points of the city after their line haul leg. On arrival, drivers may take a break, if required, and proceed to perform their respective local routes. Therefore, carriers do not collaborate at the entry to the city, nor while routing. The results are referred to as "non-collaborative entry" and "non-collaborative routing", respectively.

Next, we explain *collaborative* entry. In the proposed framework, logistics platforms exist at the entry points to the city. These platforms allow goods exchange between trucks from different carriers, and trucks of the same carrier coming from different origins. Those exchanges lead to the construction of superior local routes, resulting in higher asset utilization, and cost savings from lower route distance.

The above exchange of goods between carriers is referred to as "collaborative entry". Once local routes are dispatched, carriers can then collaborate by exchanging goods at transshipment points as outlined in Section 3.1. This is referred to as "collaborative routing", and its primary benefit, in addition to those already listed in Section 3.1, is now described.

Carriers entering from different corners of a big city have a geographical advantage by virtue of their entry location. For example, consider a rectangular city with four points of entry. A carrier entering from the north-east corner will travel less distance to serve customers located in that sector than will a carrier entering from the north-west, south-west or south-east corners. Therefore, by collaborating at transshipment points and exchanging goods, carriers will be able to spread this geographical advantage. From the above description, it can also be inferred that collaboration between the routes of different carriers entering from distinct corner points will be more beneficial than the same exchange between carriers entering from the same corner point.

For this reason, we consider all routes from a corner point to belong to the same carrier for routing purposes. This restriction acts as a good pre-processing step, reducing the running time and leading to high quality results, when using the greedy heuristic to solve the COL-VRPTW. That heuristic will indeed return collaborative scenarios between carriers entering from different corner points. Understandably, after the collaborative routes have been constructed, a post-processing step can be used to assign savings to individual carriers. Sections 3.5-3.9 will detail the solution methodology.

3.5 Transshipment Facility Location

The collaborative vehicle routing problem was described Sections 3.1 and 3.2. Collaboration requires vehicles from different carriers to exchange goods at transshipment points. Those points are facilities which do not store much inventory, if any, but allow a transfer of goods between carriers.

An interesting aspect of the COLVRP is its dynamic nature. Good facility locations depend on the availability of vehicles to serve the demand points. For LTL carriers, these demand points keep varying, which causes the transhipment locations to change as well. Therefore, a good transhipment location in week w may not be suitable in week $w+n$, where n is a positive integer. Since collaboration is driven by the resulting savings, the location of transshipment points is crucial to beneficial collaboration.

The variation in transshipment locations mean that it is impractical to build dedicated facilities. Rather, the focus should be on identifying existing urban spaces which can be used for transshipment. Crainic et al. [11, 12] and Mancini et al. [32] suggest that city bus terminals and tour bus parking lots can be used as transshipment points. These are indeed good sites for collaborative transfers, but we propose the additional possibility of transshipping at a *consignee's location*. The feasibility of transshipment at a customer's location will depend on the infrastructure of the loading dock. The idea is to give carriers and customers an incentive to collaborate. The distribution of the savings from collaboration, among carriers and consignees, is another interesting problem that stems from collaborative routing.

From a modelling perspective, the problem is to locate transshipment facilities such that the savings from collaboration is maximized. Finding those savings involves solution of the COLVRP, which is highly complex. In the literature, feedback mechanisms have been used for two-stage models, where decisions in one stage

affect decisions in the next. A feedback mechanism was used in Bookbinder and Reece [5] to solve a two stage distribution model. The first stage located facilities, and the second stage solved the resulting problem of routing from facilities to customers. In their next iteration, the second stage results from the previous iteration are used to obtain facility costs. Considering the "richness" and complexity of the COLVRP, however, a feedback mechanism would not be a desirable option in terms of solution time for us.

A good transshipment location, in terms of cost, may be infeasible in practice. Infeasibility may arise due to lack of facilities in the vicinity of the points suggested by the model. This can be overcome by using location models from network design. The term "network design" is used loosely since the decisions taken by discrete location models are always related to the opening of facilities from a candidate list. The model chooses facilities such that a single objective or multiple objectives are optimized. Many efficient solution methodologies exist, but these methods are not sufficiently robust to handle varying side-constraints and objective functions.

Considering that many subjective elements need to be considered in our location model, and accounting for temporal variations, we suggest instead a heuristic planning module for transshipment location. Our heuristic uses an adaptive quadtree search (refer to Section 2.3) which is highly flexible in handling changes to the objective function. Secondly, the search algorithm will turn out to run within a few seconds, even for large data sets. This will allow the planner (the user) to test different objective functions, before selecting the transshipment points. Finally, similar to many scheduling systems which allow user input, we alow the user to change the location of the transshipment point. It is widely accepted that planning systems with human intervention perform better. Therefore, the transshipment location model is only a guide to potentially beneficial locations, which the user can accept or change.

3.5.1 Quadtree Search

In this section, the quadtree search algorithm will be explained in detail. As mentioned before, the quadtree search locates transshipment points, which in turn affect the carriers and routes which can enter into a collaborative goods exchange. Introducing a transshipment facility is accompanied by fixed and operational costs, which need to be justified by the benefits from collaboration. The appropriateness of the chosen location can only be evaluated once the second phase is complete.

Therefore, if we wish to open three transshipment facilities, we would ideally want phase one to return the three best locations. Phase one would indeed give the best transshipment points if the heuristic evaluation function used by the quadtree search were exact, but such a function is not available. Therefore, it is used to produce a set of potential transshipment points, which are used in phase two. The savings that result from these potential sites, are then used to decide if they are to be operated or not.

As is the case for most heuristics, we thus use an *approximate* function. The output of phase one consists of clusters of customers. The distribution of customers within each cluster is used to locate the corresponding transshipment point. Clusters with desired characteristics rank higher than those that do not. For example, we may consider a good cluster to be one with customers from different carriers close together. Therefore, the evaluation function would seek clusters with a balanced ratio of customers from different carriers, and a high density of points per unit area. If n_k^p is the number of customer points of carrier k in cluster p, K is the

number of carriers, and A_p is the area of cluster p, then the evaluation function is

$$f^p = \sum_{k=1}^{K-1} \left[\delta_k^p \frac{n_k^p}{n_{k+1}^p} + (1 - \delta_k^p) \frac{n_{k+1}^p}{n_k^p} \right] + \frac{\sum_{k=1}^{K} n_k^p}{A_p}$$

$$\delta_k^p = 1, \, if \, n_k^p \leq n_{k+1}^p; \; 0, \, otherwise$$

Similarly, clusters can be characterized in various ways by using different evaluation functions. The quadtree search can incorporate these changes by modifying only the module which handles heuristic function evaluations alone.

The quadtree search procedure (Table 3.1) will now be explained. The coding was done in Java and the tree data structure from the Java Data Structure Library (JDSL) was used. The function QuadtreeSearch takes as inputs an array of evaluation function parameters $\overrightarrow{\alpha}$, the number of transshipment points or clusters to be returned, N_p, the minimum area of a cluster A_{thresh} and the minimum number of customer points in a cluster N_{thresh}^{cust}. The array $\overrightarrow{\alpha}$ contains the weights associated with each term in the evaluation function. In the present work, we always employ a convex combination, so that $\sum_i \alpha_i = 1$. The user can vary the values in $\overrightarrow{\alpha}$ to change the importance of the different terms.

Once the input values are given, the InputData() function reads the customer coordinates and other characteristics such as demand and time windows. Using spatial coordinates of customers, the InitBoundRect function initializes the bounding rectangle which encloses all the points, and assigns it to the root of the quadtree indicated as quadtree.root. In an actual data set, InitBoundRect would just involve reading coordinates of the points of entry to the urban region. For data sets that we create, this function finds the bounding rectangle and locates the entry points at its vertices.

Once initialized, the root of the quadtree is used to call the Buildtree function

```
QuadtreeSearch(α⃗ N_p, A_thresh, N^cust_thresh):
        InputData()
        InitBoundRect(minx, maxx, miny, maxy)
        Buildtree(quadtree.root)
        FindBestClusters(α⃗)
        ComputeTransshipCoords()
```

Table 3.1: Quadtree search algorithm

which recursively builds the quadtree. A quadrant is stored at every node in the quadtree and a unique "position" is used to retrieve it. When that is completed, the FindBestClusters function obtains the best N_p clusters. Finally, the ComputeTransshipCoords function determines the transshipment points within the chosen clusters. The location of those points can be calculated using a simple approach such as the demand-weighted centroid or something more complex. The methods used for siting transshipment points will be elaborated in the results section.

```
Buildtree(p)
        if(quadtree.isExternal(p)):
                if(quadrant(p).area/4 ≥ A_thresh)
                        partitionPoints(quadrant(p))
                        for each(quadrant(p).child)
                                if(child.size ≥ N^cust_thresh)
                                        child.heurval = HeurFn(child)
                                        Buildtree(Position(child))
```

Table 3.2: Function which recursively builds the adaptive quadtree

Next, the function which builds the quadtree will be described, based upon the pseudo code given in Table 3.2. Buildtree takes the "position" of the quadrant in the quadtree as input. The position can be used to extract the corresponding quadrant and information associated with it. Since every quadrant that is passed to the function will be split, the function checks if the position p is an external node. If the node is internal the function does nothing, in reality it would return

an error. On the other hand, if the node is external, the function checks if the split quadrants (children) satisfy the minimum area (A_{thresh}). If the minimum area criterion is met, then the points in the original quadrant are partitioned into the quadrants that represent the four children.

The preceeding partitioning operations takes time $O(n_p)$, where n_p is the number of points in quadrant p, and is the most costly operation in building a quadtree. Once partitioning is complete, the function checks if the number of points in each child is greater than the minimum threshold (N^{cust}_{thresh}) set by the user. If this condition is satisfied, the heuristic value of the child is computed and Buildtree is called recursively with the position corresponding to the child. The objective function of the quadtree is calculated by calling HeurFn.

We show in Appendix 5.3 that the depth of the quadtree described above is bouded by $\frac{3}{2}+\min\left[\log_2\left(\frac{l}{d}\right), 1+\log_4\left(\frac{A_{init}}{A_{thresh}}\right)\right]$. Further, in the same appendix, we also show that the quadtree has $O((d+1)n_{max})$ nodes and can be constructed in $O((d+1)n)$ time, where $n_{max} = \left\lceil \frac{n}{N^{cust}_{thresh}} \right\rceil$.

3.6 Constraint Programming Formulation of VRP

Now that the quadtree search has been explained, this section will start to detail the algorithms used to solve the VRPTWs that result from carrier collaboration, namely entry-point collaboration. The formulation most commonly used in CP based VRP solvers is presented next. This formulation is used to check feasibility in the Tabu Search and Guided local search metaheuristics. More details can be found in Kilby and Shaw [26]. The reader may wish to read Section 2.4, which gives an introduction to constraint programming, before proceeding with this section.

Let us assume that the VRP concerns n customers to be served by m available vehicles. Each point on the route is termed a "visit", as it corresponds to a visit

made by a vehicle. We define the required sets, decision variables, and constraints in Table 3.3. In this table, C is the set of customers; M is the set of vehicles and V the set of visits. Each vehicle has two visits, corresponding to its first and last visit, which in the case of the VRPTW would be at the depot. For a vehicle k its depot is denoted by visits $n+k$ and $n+m+k$. The set of all first visits is given by F, and the set of all last visits is given by L. Then, V can be written as $C \bigcup F \bigcup L$.

The decision variables of the CP model (Table 3.3) are the successor and the predecessor of each visit. Both these constraint variables are maintained, as it leads to better constraint propagation.

Sets:

$C =$	$\{1...n\}$
$M =$	$\{1...m\}$
$V =$	$\{1...n+2m\}$
$F =$	$\{n+1...n+m\}$
$L =$	$\{n+m+1...n+2m\}$

Decision Variables:

$p_i \in V \setminus L$	predecessor of visit $i \in V \setminus F$
$s_i \in V \setminus F$	successor of visit $i \in V \setminus L$
$v_i \in M$	vehicle serving visit $i \in V$
tq_i	quantity of goods on the vehicle after visiting customer $i \in V$
b_i	time at which service begins at visit $i \in V$
$cost$	total cost

Constraints:

$p_i \neq p_j$	$\forall i, j \in V$ and $i < j$	(1)
$p_{s_i} = i$	$\forall i \in V \setminus L$	(2)
$s_{p_i} = i$	$\forall i \in V \setminus F$	(3)
$v_i = v_{p_i}$	$\forall i \in V \setminus F$	(4)
$v_i = v_{s_i}$	$\forall i \in V \setminus L$	(5)
$v_{f_k} = v_{l_k} = k$	$\forall k \in M$	(6)
$tq_i = tq_{p_i} + q_i$	$\forall i \in V \setminus F$	(7)

$tq_i = tq_{s_i} - q_{s_i}$	$\forall i \in V \setminus L$	(8)
$tq_i \leq Q_{v_i}$	$\forall i \in V$	(9)
$b_i \geq b_{p_i} + t_{p_i,i}$	$\forall i \in V \setminus F$	(10)
$b_i \leq b_{s_i} - t_{i,s_i}$	$\forall i \in V \setminus F$	(11)
$E_{v_i} \leq b_i \leq L_{v_i}$	$\forall i \in V$	(12)
$tq_i \geq 0$	$\forall i \in V \setminus F$	(13)
$b_i \geq 0$	$\forall i \in V \setminus F$	(14)

Table 3.3: CP formulation of the VRPTW

Next, we describe the constraints. The difference constraint (1) ensures that a visit occurs only once. The link between the successor and predecessor variables is given by constraints (2) and (3). Equations (4), (5) and (6) ensure that all visits on a route are performed by the same vehicle, and that this vehicle's route starts and ends at a depot. The capacity of each vehicle, and the time window for every visit are maintained through path constraints. For any vehicle, the corresponding path constraints are (7) and (8), while (9) imposes the capacity Q_{v_i} of vehicle v_i on the constrained variable tq_i. Similar relations are used to model the time windows, and are given by (10), (11) and (12). The last two ensure that the constrained variables tq_i and b_i are non-negative. To complete the formulation, the objective function is defined as

$mincost = \sum_{i \in V \setminus F} d_{p_i,i} + \sum_{i \in V \setminus L} d_{i,s_i}$

where $d_{i,j}$ is the distance between visits i and j.

The above formulation with minimal modification can be used to solve a variety of routing models such as the pickup and delivery problem, open vehicle routing problem, site dependent vehicle routing problem, and the case of vehicle routing with multiple time windows. The propagation methods employed for each of the above constraints can be found in Kilby and Shaw [26].

The strengths of constraint programming include the ability to model rich con-

straints in a natural manner, its ability to handle nonlinearities with ease, and the efficient propagation methods for specialized global constraints. In the above formulation, the constraints can be written as a combination of all-different, element and path constraints, each of which has its own efficient propagation algorithm. Given a solution, the CP engine will be able to efficiently check feasibility and use its specialized algorithms to reduce the domain of the constrained variables.

For search, backtracking methods are very popular in CP. These methods are efficient at finding solutions to satisfiability problems. However, backtracking is not a suitable search method for VRPs. Therefore, we use the constraint programming framework to check for solution feasibility, and to perform constraint propagation. Search is implemented as an independent component using metaheuristics. It should also be noted that the constraint propagation helps the local search through domain reduction. The CP framework explained above is used within two metaheuristics: Tabu search and Guided local search. These methods are explained in the following two sections.

3.7 Metaheuristics

Details specific to the implementation of metaheuristics; Tabu Search (TS) and Guided Local Search (GLS), are described in this section. These metaheuristics were employed in solving the VRPTW, and the solution was used to create the COL-VRPTW instances. Refinements to the TS, and details of parameter settings for the GLS are also given here.

3.7.1 Tabu Search

Tabu search (Section 2.7.1) is a metaheuristic which uses structured learning to escape local optima and obtain high quality solutions. In this thesis, we use the standard tabu search template from Solver 6.6 from ILOG and their Dispatcher 4.6 to implement a modified version of the basic tabu search heuristic implemented in Debacker and Furnon [3]. The tabu search uses two tabu lists of fixed and equal size, one for the most recently-added arcs, and the second for the most recently-removed arcs. A move m is *tabu* if the sum of the number of removed and added arcs in the tabu list is greater than a threshold limit. We use the same threshold limits as in [3], and these values are reproduced in Table 3.4.

Operator	Arcs	Threshold
Cross	4	3
Exchange	8	6
Relocate	6	5
Or-opt	6	5
2-Opt	≥ 3	3

Table 3.4: Thresholds for tabu moves

The Cross, Exchange, Relocate, Or-opt, and 2-Opt neighborhoods (Kindervater and Savelsbergh [27]) are used for search, and an insertion neighborhood for diversification. The diversification method used has two phases. Firstly, a solution is declared as a local optimum if the search has N_{noimp} non-improving iterations. The number of non-improving moves is stored in a counter, C_{noimp}. Once a local optimum is detected, the tabu tenure, t, is decreased by a fixed amount δ_1. This serves to diversify the search. If an improving move is then found, C_{noimp} is reset to zero. Otherwise, the tabu search continues to reduce t, until $C_{noimp} \geq 2N_{noimp}$. At this stage, we use a probabilistic diversification method, motivated by the WALKSAT (Russel and Norvig [39]) algorithm. The idea is to randomly reinsert a fixed number

of customers to another route from their current one.

```
TabuSearch(N_imp,N_noimp,t,N_insert,δ1,δ2)
C_noimp = 0
C_imp = 0
tenure = t
s = InitSolution()
s = LocalSearch(s)
s* = s
while not STOP
        M = RankMoves(s)
        moved = false
        while |M|
                m = first(M)
                if not Tabu(m)
                        Perform(m)
                        moved = true
                        ModifyTabuList(m)
                        if f(s) < f(s*) s* = s
                                C_imp = C_imp + 1
                                C_noimp = 0
                        else
                                C_noimp = C_noimp + 1
                                C_imp = 0
                        if C_noimp > 2N_noimp
                                s = Diversify(N_insert,s)
                        else C_noimp > N_noimp
                                SetTenure(t - δ1)
                        if C_imp > N_imp
                                SetTenure(t + δ2)
```

Table 3.5: Tabu search statement

First, a customer c is chosen at random and removed from the solution. The current vehicle serving that customer is stored in v. Then we add the constraint

: $v_c \neq v$. This states that the vehicle which serves the customer c cannot be the current vehicle v. Now we perform an insertion operation. This is repeated a fixed number of times, before restarting the tabu search. Finally, we remove all constraints added during diversification. Our results indicate that this scheme produces superior results in comparison to the basic tabu search. This is not surprising, as randomized algorithms have performed extremely well in satisfiability problems, where local optima are escaped by flipping variables in clauses. Though our diversification method was motivated by randomized algorithms for the satisfiability problem, probabilistic diversification schemes have been used with great success in the VRPTW by Rochat and Taillard [36]. For intensification of the tabu search,

Diversify(N_{insert},s)
$s^{'} = s$ for i = 1 to N_{insert} visit := RandomVisit($s^{'}$) v := Vehicle(visit,$s^{'}$) $s^{'}$:= RemoveVisit(visit,$s^{'}$) addConstraint($v_{visit} \neq v$) $s^{'}$:= insert(Visit,$s^{'}$) RemoveAllConstrants() return $s^{'}$

Table 3.6: Statement of *Diversify*

we reverse the first step of the diversification method by increasing the tabu tenure if N_{imp} consecutive improving iterations have been encountered. A summary of the overall algorithm is given in Table 3.5. In that statement, *RankMoves(s)* returns all legal moves in increasing order of their cost. *First(M)* returns the first element of the set M. The condition *STOP* represents the stopping criteria for TS, which was set to be a maximum number of iterations. The *Diversify* method implements the diversification scheme already explained. The details are given in Table 3.6.

3.7.2 Guided Local Search

Guided local search was introduced in Section 2.7.2. There and here, we use the standard GLS template from ILOG Dispatcher 4.6. The weight of the penalty term (λ) in the objective function (Eq 2.1), is the only parameter that needs to be tuned. For VRPTWs of size less than one hundred customers, the best value of λ varied, but for larger problems, a value of 0.2 gave the best average case performance. This value was used for all the numerical results that we report in the next chapter.

For small data sets, our modified TS was consistently better than GLS. Mainly because of the random nature of the diversification scheme used in TS, the deterministic GLS sometimes outperformed the modified TS. Therefore, in certain cases GLS, rather than our enhanced version of TS, was used to obtain the VRPTW results. In particular, for larger data sets, we employed the least-cost solution from the two methods.

3.8 Greedy Local Search for Collaborative Routing

So far we have covered the heuristics required to solve VRPTWs. However, collaboration between carriers has two stages: Collaboration at the entry to a city, which involves the solution of VRPTWs, and then collaborative routing. For the latter form of collaboration, which involves the solution of the COL-VRPTW, we use an *integrated* greedy local search heuristic. Our method is integrated since we use an exact optimization method within local search to return a set of feasible moves ("Super Move"). This integration will be explained in this section. To define a local search method (Section 2.5.1) we need to specify a local improvement strategy, a neighborhood, and a stopping criteria. The design in light of these three decisions

will be explained below.

We first describe the three *moves* that define the search neighborhood. A move is an operator, governed by a set of rules, which allows a tractable neighborhood to be described. This in turn has great impact on the search, as complex moves may lead to larger neighborhoods, but may also be prohibitively time consuming to explore. For example, consider the move operators, 2-opt and 3-opt (Kindervater and Savelsbergh [27]). A 2-opt neighborhood can be searched very quickly and is used in almost all VRP local improvement schemes. On the other hand, 3-opt is considerably better at improving the solution but is time consuming, and therefore a more restricted version called Or-opt is used instead.

Heuristics for VRP with transshipment tend to use traditional move operators to handle transshipment. In contrast, we use *transshipment-specific* moves and consider transferring a sequence of consecutive customers from one route to another. The route from which customers are transferred is called the load giver (LG) route, and the destination route for the transferred sequence is referred to as the load taker (LT) route. For all move operators, the following decisions need to be taken:

1. The customer after which the LT truck leaves the route to visit the transshipment point. The same decision has to be taken for the LG as well.

2. The customer that is first visited after the LT truck returns to its route from the transshipment point. Similarly for the LG.

3. The sequence of customers that are transferred from the LG route to the LT route.

Our three move operators differ in how these decisions are made, and hence define exclusive neighborhoods. We make the assumption that the transferred sequence is inserted in between *successive customers* in the LT route. Our test

runs showed that allowing sequence splitting and multiple insertions make the move extremely complex and prohibitively time consuming. All three moves consider the transfer of customers between a single route from a load taker and another route from the load giver.

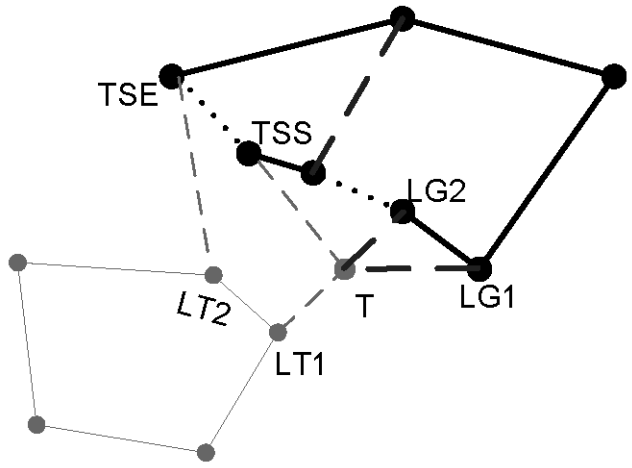


Figure 3.9: Transshipment heuristic sequence move 1

First let us consider Sequence Move 1 (SM1) represented in Figure 3.9. In this figure and other move representations, dotted arcs represent a sequence of customers, solid lines together with dotted arcs represent each carrier's route before the move, and dark-dashed and light-dashed arcs are those added to the load giver and load taker, respectively, during the move. We assume that the load giver leaves the route at customer LG1, visits the transshipment point T, and returns to the route at LG2, which was the initial successor of LG1. In other words, T is inserted in between consecutive customers LG1 and LG2. The start and end of the transferred sequence is identified by the customers labelled Transshipment Sequence Start (TSS) and Transshipment Sequence End (TSE) respectively.

The load taker leaves its route at customer LT1 and visits the transshipment point T. From T, the truck proceeds to serve the customers in the sequence (i.e. $\{TSS...TSE\}$), before returning to its route at LT2. Therefore, the nodes $T \bigcup \{TSS...TSE\}$ are inserted between consecutive customers LT1 and LT2.

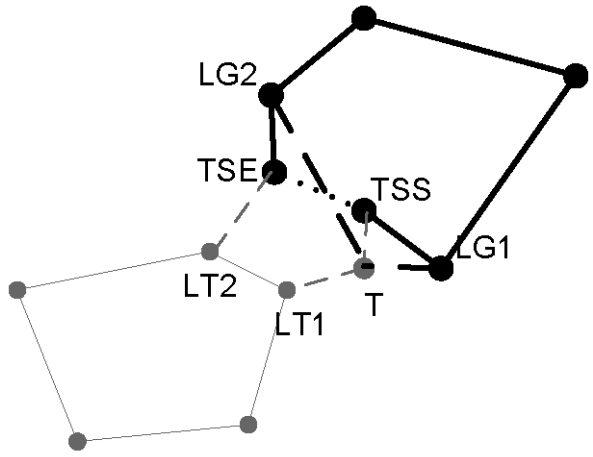


Figure 3.10: Transshipment heuristic sequence move 2

In Sequence Move 2 (SM2) (Figure 3.10), the load taker's operations are identical to SM1, but now the transferred sequence is removed from between LG1 and LG2. Therefore, LG1 and LG2, unlike in SM1, are not successive customers.

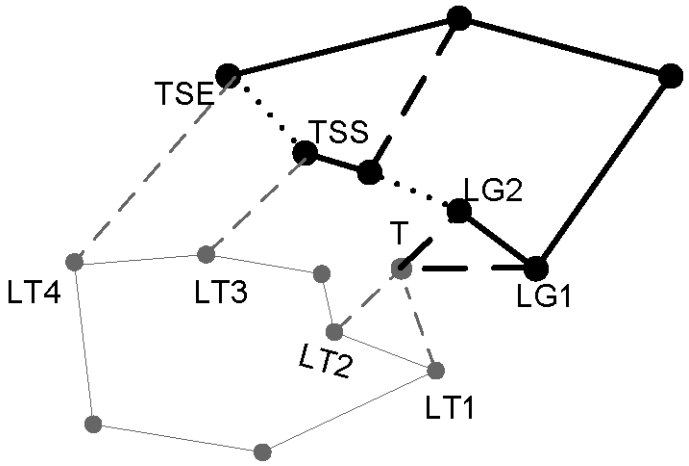


Figure 3.11: Transshipment heuristic sequence move 3

In SM1 and SM2, the load taker visited TSS immediately after the transshipment point. Sequence Move 3 (SM3) considers the situation where the load taker returns to the route at LT2, after receiving loads corresponding the transferred sequence (i.e. $\{TSS...TSE\}$), but before actually serving those customers (Figure 3.11). To deliver the loads collected at T, the load taker deviates from the route at LT3, a customer later in the route, and serves the customers $\{TSS...TSE\}$, before

return to its own route at LT4.

SM3 is understandably the most complex move, as additional decisions regarding LT3 and LT4 have to be made. All the preceding moves preserve the *direction* of the transferred sequence. In this regard, our moves have some similarity to Or-opt. A summary of the edges removed and added when using each move is given in Table 3.7. In VRP algorithms, the number of edges added and removed are used to indicate the complexity of a move. From this and our empirical tests, it was seen that SM2 was least time consuming, while SM3 was an expensive operator to use in terms of computational time.

Operator	Edges Removed	Edges Added
SM1	4	6
SM2	3	5
SM3	5	7

Table 3.7: Edges added and removed when using SM1, SM2 and SM3

The search takes the following as input: VRPTW routes corresponding to each entry point to the city, transshipment points output by the quadtree search algorithm, and the cluster of customers associated with each transshipment point. A step-wise explanation of the algorithm is presented below.

1. For each transshipment point $tp \in T$, propagate time window feasibility constraints for all customers in the cluster. This is done by keeping a feasibility record for each customer $c \in Cluster_{tp}$ for all $tp \in T$. That is, we check if adding tp in between c and its predecessor, or c and its successor violates the time window constraints. A unique tag is attached to c which can be used to check if both or only one of the above cases is feasible. c is removed from $Cluster_{tp}$ if both the preceding insertion cases are infeasible.

2. Sort the clusters resulting from Step 1 in *decreasing* order of size. Within each cluster, sort the customers in decreasing order of time window slack, $l_i - (b_i + s_i)$ for all $i \in C$

3. Call the transship module (Table 3.8), which returns one of the following: (a) The first feasible transshipment opportunity in the neighborhood; (b) the best opportunity, or (c) the set of transshipment opportunities that maximize savings. The above setting is controlled by a parameter called *type*. If type equals "greedy set", then an optimization model is called to find those transshipment opportunities that maximize the savings. If type is instead "first accept" or "best accept", the module returns the first feasible neighbor or the best feasible neighbor, respectively.

4. Update the routes, using the the set of chosen transshipment opportunities. This is when we actually make the move to a new solution.

In the preceding description, we only consider those customers in the same cluster as the transshipment point for collaboration. This is equivalent to inherently using a radius parameter (Section 3.3.1). The transship module used in the above description will now be explained. This module takes as input the type of search. As explained in Step3, the value of the parameter *type* can be first feasible, best feasible, or greedy set. For each cluster $tp \in T$, we first check if its size, represented by *cluster.size()*, is at least one. This is necessary to ensure that at least one unique customer which represents a load taker, is available.

Next, we check if the cluster size is at least two, and if this is the case, we assign SM1 and SM3 to variable *Moves*. This variable defines the operators to be used in constructing the neighborhood for search. After time propagation, the preceeding condition has to hold for moves SM1 and SM3, because we need a unique load taker and load giver in those cases. Unfortunately, this is not possible for SM3. In

```
Transship( Type )
for all tp ∈ T
    if Cluster.size() > 0
        if Cluster.size() > 1
            Moves = SM1 + SM3
        else
            Moves = SM2
        for LT ∈ Cluster_tp
            LTFeas = FeasibleLT(LT)
            LGSet = ChooseFeasibleLG(LT)
            for LG ∈ LGSet
                LGFeas = FeasibleLG(LG)
                Collaborate(Moves, c, LG, LTFeas, LGFeas, Type)
```

Table 3.8: Statement of the *Transship* module

that case, the location of LG2 is decided during the move itself, and therefore time propagation cannot be used to eliminate infeasible load givers before the move.

Next, we iterate over all nodes in $Cluster_{tp}$. For each node LT, we retrieve its feasibility status using the tag assigned to it during time window propagation (Step 1). Then, given LT, we find the set of all possible load givers using the *ChooseFeasibleLG(LT)* function, which searches for the set of all nodes in the same cluster which also belongs to a different carrier than LT. This restriction is added to ensure that only inter-carrier collaboration is considered. Once that is done, the information is passed to the *Collaborate* function which actually makes the move. This completes the transship module. A description of the optimization module used in step 3 is given below.

Recall that this module is needed only if the parameter *type* is equal to greedy set in the transship module input. Assume that we are at a solution s. We define a

transshipment opportunity to be a move from the current solution to a new feasible solution, which also results in distance savings (i.e. it represents an improving feasible move). We define TO to be the set of all possible transshipment opportunities. Each element in this set will contain the information required to make the changes and move to the new solution it represents.

When searching for the best improving feasible move, we explore the neighborhood of a solution (or a subset of the neighborhood) before declaring that solution to be the best. During exploration, we encounter many improving feasible moves that might not be used. In the greedy set case, we try to combine those opportunities in an attempt to more efficiently use the information gathered during neighborhood exploration. To combine these moves, we use a mathematical programming model which greedily selects the savings-maximizing subset of moves which, when used on the current solution, will lead to a feasible solution. We refer to this combination of feasible moves as a "super move." Understandably, not all combinations of moves are feasible. Therefore, we use the optimization model in Table 3.11 to find the greedy set.

Parameters:

Sav_{to}	Savings if transshipment opportunity to is used
C^{lt}_{to}	Load taker carrier associated with to
R^{lt}_{to}	Load taker route associated with to
$C1^{lt}_{to}$	Customer LT1 associated with to
$C2^{lt}_{to}$	Customer LT2 associated with to
$rC3^{lt}_{to}$	rank of customer LT3 in route R^{lt}_{to}
$rC4^{lt}_{to}$	rank of customer LT4 in route R^{lt}_{to}
CR^{lg}_{to}	Load giver carrier associated with to
R^{lg}_{to}	Load giver route associated with to

$C1_{to}^{lg}$ Customer LG1 associated with to

$C2_{to}^{lg}$ Customer LG2 associated with to

rS_{to} rank of TSS in route R_{to}^{lg}

rE_{to} rank of TSE in route R_{to}^{lg}

ID_{to} ID of the move operator associated with tpo (i.e. SM1, SM2 or SM3)

Sets:

S_{to}^1 $\{\ to' \mid C_{to'}^{lt} = C_{to}^{lt}\ and\ R_{to'}^{lt} = R_{to}^{lt}\ and\ to', to \in TO\ \}$

S_{to}^2 $\{\ to' \mid C_{to'}^{lg} = C_{to}^{lt}\ and\ R_{to'}^{lg} = R_{to}^{lt}\ and\ to', to \in TO\ \}$

S_{to}^3 $\{\ to' \mid to' \in S_{to}^2\ and\ ID_{to'} \neq SM3\ \}$

S_{to}^4 $\{\ to' \mid to' \in S_{to}^1\ and\ (\ C1_{to'}^{lg} \neq C1_{to}^{lg}\ or\ C2_{to'}^{lg} \neq C2_{to}^{lg}\ )\ \}$

S_{to}^5 $\{\ to' \mid to' \in S_{to}^1\ and\ to' \neq to\ and\ (rE_{to} \geq rE_{to'} \geq rS_{to}$

$or\ rE_{to} \geq rS_{to'} \geq rS_{to}\ or\ (rS_{to'} \leq rS_{to}\ and\ rE_{to'} \geq rE_{to}\ )\ )\ \}$

S_{to}^6 $\{\ to' \mid to' \in S_{to}^2\ and\ (\ rE_{to'} = rC3_{to}^{lt}\ or\ rS_{to'} = rC4_{to}^{lt}$

$or\ rS_{to'} = rC3_{to}^{lt}\ or\ rE_{to'} = rC4_{to}^{lt})\ \}$

S_{to}^7 $\{\ to' \mid to' \in S_{to}^2\ and\ ID_{to'} = SM3\ and\ (\ C1_{to}^{lt} \neq C1_{to'}^{lg}\ or\ C2_{to}^{lt} \neq C2_{to'}^{lg}\ )\ \}$

Model:

$$T_{to} = \begin{cases} 1, & \text{if transshipment opportunity } to \text{ is chosen;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\textbf{max} \sum_{to \in TO} Sav_{to} T_{to} \tag{1}$$

$$\sum_{to' \in S_{to}^1} T_{to'} \leq 1 \qquad \forall to \in TO \tag{2}$$

$$\sum_{to' \in S_{to}^5} T_{to'} \leq |S_{to}^5|\,(1 - T_{to}) \quad \forall to \in TO \tag{3}$$

$$\sum_{to' \in S_{to}^4} T_{to'} \leq |S_{to}^4|\,(1 - T_{to}) \quad \forall to \in TO \tag{4}$$

$$\sum_{to' \in S^3_{to}} T_{to'} \leq |S^3_{to}|\,(1 - T_{to}) \quad \forall to \in TO \text{ and } ID_{to} \neq SM3 \quad (5)$$

$$\sum_{to' \in S^6_{to}} T_{to'} \leq |S^6_{to}|\,(1 - T_{to}) \quad \forall to \in TO \text{ and } ID_{to} = SM3 \quad (6)$$

$$\sum_{to' \in S^7_{to}} T_{to'} \leq |S^7_{to}|\,(1 - T_{to}) \quad \forall to \in TO \text{ and } ID_{to} = SM3 \quad (7)$$

Table 3.11: *Greedy set* optimization model

Table 3.11 contains the parameters, sets, the objective function and constraints of the optimization model. The rank of a customer is the index of that customer in its route, starting with the depot as zero. The binary decision variables T_{to} represent whether a transshipment opportunity is chosen or not. The objective is to maximize the savings from the chosen TOs. Constraint (1), ensures that each chosen TO has a unique LT carrier and route. The fact that no transferred sequences can overlap is captured by the second inequality. Constraint (3) ensures that multiple sequences can be transferred from a load giver, if it has the same customers as LT1 and LT2. The condition that the same carrier route cannot be both a load taker and load giver, for moves other than SM3, is represented by constraint (5). If a move of type SM3 is used, then if it acts as both a load taker and load giver, the transferred sequence should not coincide with LT3 and LT4. This condition is encoded in (6). (7) enforces the condition that if a carrier and route are involved in an SM3 type move as both a load taker and a load giver, then LT1 must equal LG1 and LT2 must equal LG2. We would like to highlight that there exists considerable redundancy in the formulation, but our tests showed that the model is actually solved to optimality within a few seconds.

3.9 Summary of the Overall Solution Procedure

The algorithms employed to solve the COL-VRPTW have been presented in this chapter. In the following, we summarize the steps to solve the COL-VRPTW with references to the algorithms used.

1. Create l_1 random VRPTW problems for each corner of the rectangular region used to model the city. The problems are generated as described in Appendix 5.3 and Section 4.2.

2. Solve each VRPTW generated in step 1, using tabu search (Section 3.7.1) or guided local search (Section 3.7.2)

3. Group l_2 problems, where $l_2 \leq l_1$ from each corner and create a larger VRPTW instance.

4. For each corner, solve the VRPTW created in the previous step using tabu search or guided local search.

5. Create the COL-VRPTW problem using the four VRPTWs from step 3. In other words, the COL-VRPTW is made up of four VRPTWs, each corresponding to a unique corner of the rectangle.

6. Use the quadtree search algorithm described in Section 3.5.1 to locate transshipment points and define clusters

7. Solve the collaborative routing problem using the greedy local search from Section 3.8. Use the routes from step 4 as an initial solution.

Now we proceed to explain each step in the above summary. The problem of collaboration starts at logistics platforms located at the boundary of the urban region. We take this to be a rectangle with a logistics platform at each corner.

Inbound loads carried by multiple carriers arrive at these facilities after inter-city line haul.

In step 1, we assume that logistics platforms do not exist. This corresponds to the *non-collaborative entry* case. In this step, each VRPTW generated for a corner corresponds to a carrier's delivery schedule from that point of entry. The routes obtained in step 2 are therefore the non-collaborative entry routes.

In step 3, we consider collaboration. Now we assume that logistics platforms are present at each corner. Therefore, carriers can exchange goods after the inbound leg, before planning their intra-city routes. The loads or orders that are exchanged depend on the distance savings produced by the resulting collaborative local route. This is determined from the solution to a VRPTW instance, which is created by combining l_2 VRPTWs that correspond to a particular corner. An instance of this type is generated in step 3, and is solved in step 4. The initial solution to the latter step is provided by the non-collaborative routes from step 2. That idea is similar to the warm-start technology in mathematical programming, where a solution to a smaller problem is used as a feasible starting solution to the larger one.

A comparison between the collaborative entry routes from step 4, and the non-collaborative entry routes from step 2, will indicate which loads are to be exchanged between carriers. A similar comparison, in terms of distance, time and vehicles saved, can also be used to evaluate the benefits of collaborating at a logistics platform. Now that the individual collaborative-entry routes have been generated, the collaborative routing problem can be solved. To do this, the COL-VRPTW (Section 3.2) instance has to be generated. As described in Section 3.4, only carriers entering from different corners of the city can collaborate during routing. To achieve this, we assign the same carrier number to all routes starting from the same corner when generating the COL-VRPTW in Step 5.

Once the problem has been generated, we use the quadtree search to locate the transshipment points, and define the cluster of customers assigned to that point in Step 6. Now we have all the required information to solve the collaborative routing problem. That solution is obtained using a greedy local search heuristic in Step 7. This completes the description of the solution procedure, for which results are provided in the next chapter.

Chapter 4

Results and Evaluation

We present computational results from three algorithms (Tabu Search, Quadtree Search and Greedy Local Search), and interpret the results in light of our proposed framework for carrier collaboration. At each stage of our solution exposition, we refer to the overall solution procedure outlined in Section 3.9. Therefore, the reader is encouraged to review it. This chapter is structured as follows: Section 4.1 presents results from our modified tabu search for constructing vehicle routes on Solomons [41] VRPTW benchmark. Results from entry-point collaboration are given in Section 4.2. Finally, Sections 4.3 and 4.4 contain results from the quadtree search algorithm and the greedy local search for collaborative routing, respectively. For all computations to follow we use a speed of 30 Km/hr to link time and distance (Appendix 5.3), except for Solomon's benchmark (Section sec:BenchmarkResults) which assumes that distance travelled equals travel time.

4.1 Benchmark Results

Results of using our modified Tabu Search algorithm on Solomon's [41] benchmark data set are presented here. The data sets in [41] are made up of six classes: C1,

C2, R1, R2, RC1 and RC2. Each problem has 100 customers. We assumed that unlimited vehicles are available and all vehicles in a given problem have uniform capacity. Time windows on customers and vehicles are also present. The C problems have clustered customers and are relatively easy to solve. The R problems are randomly generated and have no inherent spatial structure. The RC problems are a mixture of random and clustered customers. Further, C1, R1 and RC1 are short horizon problems. There solutions require more vehicles, and the number of customers per route is small. In contrast, problem sets C2, R2 and RC2 have a long horizon and require fewer vehicles. TS has four main parameters which have

Parameter	Value
N_{imp}	10
N_{noimp}	10
t	5
N_{insert}	30
δ_1	5
δ_2	3

Table 4.1: Tabu search parameter values

to be tuned. They are: the number of improving moves before an increase in tenure (N_{imp}), number of non-improving iterations before a decrease in tenure (N_{noimp}), the starting tenure (t), the number of insertions performed during diversification (N_{insert}), reduction in tenure after N_{noimp} non-improving moves (δ_1), and increase in tenure after N_{imp} improving moves (δ_2). After rather exhaustive computational testing on C1, R2 and RC1, these parameter were set to the values shown in table 4.1. Tuning the values of an algorithm is similar to manually training it. To obtain the true performance, the training set should only be a subset of all the problems being solved. Therefore, we used C1, R2 and RC1 as our training set, and C2, R1 and RC2 as the test set.

Our implementation of TS minimizes the route distance and not the number of vehicles. Further, parameters are tuned with the goal of good performance on large data sets. We use DeBacker and Furnon [3] (DF) and Tan et al. [44] (TLZ) for our comparison as these are the only two tabu search implementations which minimize distance.

Method	R1	R2	C1	C2	RC1	RC2	CNV/CTD
TS	**13.83**	5.5	**10.00**	3.00	14.00	6.63	**500**
	1204.47	**875.70**	**828.38**	591.66	1398.34	**1020.57**	**54751**
DF	14.17	5.27	10.00	3.25	14.25	6.25	508
	1,214.86	930.18	829.77	604.84	1,385.12	1,099.96	56,998
TLZ	13.83	3.82	10.00	3.25	13.63	4.25	467
	1,266.37	1,080.24	870.87	634.85	1,458.16	1,293.38	62,008

Table 4.2: Results on Solomon's VRPTW data set

Table 4.2 contains the results of our TS implementation using ILOG Solver 6.6 and ILOG Dispatcher 4.6. The first and second rows for each method give the average number of vehicles used and the average route distance, respectively, for each data set. The last column gives the Cumulative Number of Vehicles used (CNV) and the Cumulative Travel Distance (CTD) in the first and second rows, respectively, over all data sets.

Our modified tabu search outperforms both DF and TLZ. DF has a shorter average route distance than ours only in RC1. In all other data sets, TS finds routes of lower distance. We also either equal or do better in the number of vehicles used for the R1, C1 and C2 data sets.

The stopping criteria was set to be a maximum iteration limit of thousand. The results on Solomon's benchmark are certainly encouraging. However, we would like

to emphasize that when compared to Debacker et al. ([3]) which is the closest implementation to ours, the number of iterations used was much less, in fact one-third. This reduced number of iterations was because our design goal was to to obtain high quality results very quickly, as some of the random data sets to be described shortly are quite large and more difficult to solve. In the following sections, we evaluate the results from our three-phase heuristic for carrier collaboration.

4.2 Vehicle Routing Problem with Time Windows: Collaboration at Entry Points

In this section, we describe the VRPTW instances that were created and solved in Steps 1 - 4 of the solution procedure outlined in Section 3.9. Generation of the VRPTW data sets is detailed in Appendix 5.3.

In the VRP literature, most papers evaluate a known benchmark or provide results for one combination of parameters. The latter is because running times are usually large for these problems. Even those papers which solve multiple random instances report only the *mean* of their computational results. This can be quite misleading, since the corresponding distribution can have a large dispersion, which would make the mean a very poor estimate of the reported statistic. To overcome this, we generate 40 sample points for each parameter combination (i.e. l_1 from Step 1 of Section 3.9 equals 10). This consequently involved solving 1440 VRPTWs for every value of n in Step 2, and 360 problems for each n in Step 4. The parameters used to define these random instances will be explained next.

We use Toronto, which has an area of 1749 km^2 (www.statcan.ca), as our test city. We also assume the city to be bounded by a rectangular region, and locate the logistics platforms at its corners. These platforms can be used to transfer loads

between carriers, once the inbound loads arrive and before those outbound are dispatched (Figure 2.2). Three cases are evaluated for the "region type": (i) the region is a square [i.e. length (l) = breadth (b)], (ii) the region is a rectangle with $l = 1.5b$, and (iii) $l = 2b$. The preceding cases will be referred to as RType equals 1, 2 and 3 respectively. A test case is characterized by the following parameters: number of customers per instance for the no collaboration case (n), number of customers per instance for the collaboration case (N), the region type ($RType$), percentage of customers with time windows ($\%TW$), the tightness of time window width (TWw) and the percentage of customers with low demand ($\%LD$). The different values of these parameters are given in Table 4.3.

Parameter	**Values**				
n	25	50	75	150	300
N	100	200	300	600	1200
$RType$	$l = b$	$l = 1.5b$	$l = 2b$		
$\%TW$	50	75	100		
TWw	2	5			
$\%LD$	80	95			

Table 4.3: Parameter values used to create the VRPTW test set

Recall from Section 3.9, that we first solve an individual VRPTW for each carrier at the four corners of our region. This corresponds to the Non-collaborative Entry Case (NEC). For the Collaborative Entry Case (CEC), we combine four individual problems from each corner to create a problem of size $N = 4n$ (i.e. l_2 from Step 3 of Section 3.9 equals 4). These problems are solved using TS or GLS as in the NEC, but now we use the non-collaborative routes of individual carriers as the starting solution. Comparison of these two cases will give the benefits of collaboration at the logistics platforms (Section 3.4) located at the entry points

A summary of results is given in Table 4.4 for $N \in \{100, 200, 300\}$. For each

statistic, we provide the mean (μ) and Coefficient of Variation (CV), except for one case where the standard deviation (σ) is given. For a given value of N, the mean corresponds to the average over all parameter combinations shown in Table 4.3. This is important, as real-life problems encountered by carriers are random, and could correspond to any of these parameter combinations or others.

CV measures the width or dispersion of a distribution. If the CV is less than one, the distribution is said to have a low dispersion. The closer CV is to zero, the better is the sample mean at estimating the actual value. Note that, even CV may not be the best measure, but this thesis takes a step to report more than just the mean. An ideal measure would be distribution-free confidence intervals, which are more complicated, as they require a substantially higher number of samples. However, this is an extension of the current work that is being pursued.

Table 4.4 contains the values of the total distance Savings (Sav) in kilometers, Average distance Savings per Vehicle (ASV), the average percentage Increase in Asset Utilization (IAU), the number of Vehicles Saved (VS) and the Route Time Reduction (RTR) in minutes. We give the standard deviation for RTR because the routing time *increased* in certain samples. When a distribution has negative values, the CV is less meaningful. An increase in routing time is acceptable, as route time minimization was not an objective in the VRPTWs that were solved.

N	**Sav**		**ASV**		**IAU**		**VS**		**RTR**	
	μ	$\frac{\sigma}{\mu}$	μ	$\frac{\sigma}{\mu}$	μ	$\frac{\sigma}{\mu}$	μ	$\frac{\sigma}{\mu}$	μ	σ
100	212.11	0.18	4.85	0.22	5.69	0.19	1.4	0.36	341.61	222.74
200	289.24	0.16	4.20	0.24	3.75	0.25	1.7	0.30	306.49	368.85
300	387.63	0.26	4.17	0.31	2.92	0.17	2.1	0.26	93.55	541.24

Table 4.4: Results for Mean and Coefficient of Variation $\left(CV = \frac{\sigma}{\mu}\right)$ for the CEC

The coefficients of variation for Sav, ASV, IAU and VS are all less than 0.5,

which indicates that the mean is a representative value. Savings increase with N, since more opportunities exist to build better loads. ASV and IAU decrease with N. These parameters are averaged over the vehicles, therefore, the decrease demonstrates that the improvement in ASV and IAU do not scale with the increase in number of vehicles. However, the average utilization increases by approximately 6% and 4%, while the number of vehicles used decreases by 1.4 and 1.7 on average, for $N = 100$ and $N = 200$. RTR on the other hand has a high value of σ, which in the cases of N equals 200 and 300, respectively, is greater than μ. This corroborates our previous statement that the mean alone is not sufficient to decide whether the results are beneficial.

N	**Sav**			**ASV**			**RTR**		
	%	Q1	Q3	%	Q1	Q3	%	Q1	Q3
100	14.3	183.20	235.96	6.92	4.07	5.59	3.60	153.02	503.66
200	9.70	253.53	322.41	6.03	3.24	4.74	1.55	64.53	503.05
300	9.10	317.77	477.69	6.16	3.33	4.53	0.03	-324.08	404.26

Table 4.5: Percentage improvement and quartile results for the CEC. The quartiles have the respective dimensions, km, km and min, for Sav, ASV and RTR.

To get a better sense of the benefits from entry-point collaboration, we provide the first quartile (Q1) and third quartile (Q3) of each distribution in Table 4.5. Q1 and Q3 respectively represent the values below which 25% and 75% of the sample observations fall. In particular, for RTR in the case $N = 100$, note that even though the standard deviations were high, we save over 2.5 hours for 75% of the observations and more than 8.5 hours for 25% of the observations. Results for $N = 300$ are striking as we reduce RTR by over 1 hour for 75% of the cases, and by 8.5 hours for 25% of the observations, even though the value of σ was about five times that of μ.

Further, the results in Table 4.5 show that the total route distance reduced by

approximately 14%, 10% and 9%, for $N = 100$, 200 and 300, respectively, which is a considerable saving. Over 183 kms for $N = 100$ and more than 317 kms for $N = 300$, each case for 75% of the observations. On average, the route distance of each vehicle also decreased by about 6% in each of the above cases.

N	**Sav**	**ASV**	**IAU**	**VS**	**RTR**
100	227.51	4.85	7.00	1.28	184.59
200	310.98	3.69	5.00	2.25	435.72
300	489.54	4.36	3.00	3.33	395.89
600	974.76	6.51	2.00	1.14	-471.25
1200	1107.56	3.74	1.17	1.92	1673.97

Table 4.6: Results for mean and Coefficient of Variation for the CEC, where $RType = 1$, $\%TW = 75$, $TWw = 2$ and $\%LD = 95$

Average results for the case where $RType = 1$, $\%TW = 75$, $TWw = 2$, $\%LD = 95$ and $N \in \{100, 200, 300, 600, 1200\}$ are given in Table 4.6. This set of parameters was chosen as one most likely to occur in practice. We do not report the coefficient of variation as it was below 0.1 for all the variables indicated. Distance savings increase with N, as shown in Table 4.6 and in Figure 4.1. IAU decreases with increasing N. This trend makes sense as the graph of the number of vehicles used as a function of N (Figure 4.1) has a slope of 2. Therefore, for IAU to show an increasing trend, its value should more than double when there is a two-fold increase in N. The results for VS showed no particular trend, though its values suggest that collaboration is beneficial.

Tables 4.7 and 4.8 contain more granular results, which show the effects of variations in Rtype and PTW, respectively. The distance savings increase with an increase in the number of customers having time windows. This trend is because, non-collaborative problems which are highly restricted can benefit more from collaboration. The preceding result highlights exactly why trucking companies should

RType	1			2			3		
N	**Sav**	**VS**	**RTR**	**Sav**	**VS**	**RTR**	**Sav**	**VS**	**RTR**
100	251.88	1.6	395.57	208.06	1.4	273.39	176.40	1.3	355.87
200	337.21	1.5	420.48	280.54	1.7	110.13	249.96	1.8	388.84
300	516.80	2.6	114.76	340.09	2.1	287.34	306.01	1.7	-121.46

Table 4.7: Results for the CEC with variations in RType. The values of Sav, VS and RTR in each column are averages over all observations where RType is equal to the corresponding value in row 1

collaborate. A surprising result, however, is that there is a clear reduction in distance savings (Sav) when the shape of the region deviates from a square (Table 4.7).

We conjecture, based upon analysis of the actual routes, that this is due to the following. As a region becomes skewed, there is an increase in the largest distance of any customer from the depot. These customers are served near the end of their route, and closer to their latest starting times. There will thus be only a few other routes with which such a route could feasibly collaborate, and even if they did collaborate, there would likely be little to no distance savings. The latter is because these would be fewer customers on an elongated route through a skewed region. Our reasoning is also supported by the trend of decreasing VS with RType for $N = 100$ and 300.

PTW	50			75			100		
N	**Sav**	**VS**	**RTR**	**Sav**	**VS**	**RTR**	**Sav**	**VS**	**RTR**
100	220.99	1.4	328.48	204.52	1.4	242.16	235.93	1.4	485.63
200	-	-	-	267.49	1.7	309.66	310.98	1.6	303.31
300	-	-	-	364.79	2.3	439.79	410.47	2.0	-252.69

Table 4.8: Results for the CEC with variations in PTW. The values of Sav, VS and RTR in each column are averages over all observations where PTW is equal to the corresponding value in row 1

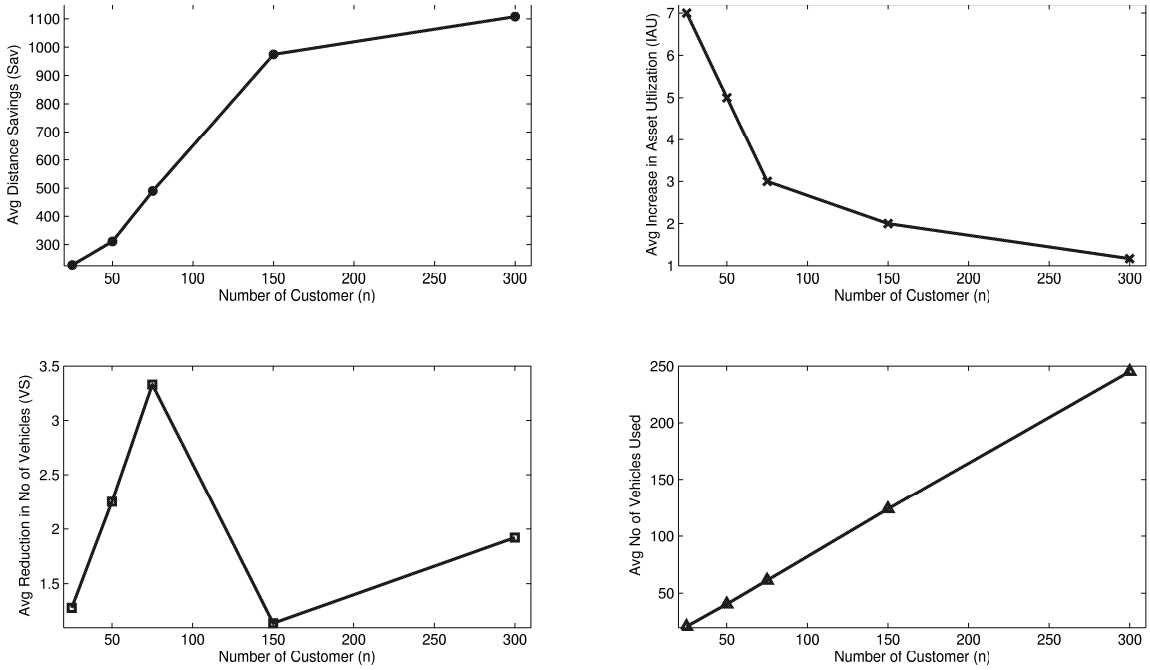


Figure 4.1: Graphical summary of CEC results when $RType = 1$, $\%TW = 75$, $TWw = 2$ and $\%LD = 95$

The effects of variations in TWw is given in Table 4.9. As TWw becomes smaller, which corresponds to tighter time windows, the distance savings decrease as expected. For RTR, we see that for $N = 200$ and 300, more routing time is saved when time windows are tighter. This is because problems with tighter time windows lead to greater waiting times, which can be considerably reduced through collaboration.

TWw	**2**			**5**		
N	**Sav**	**VS**	**RTR**	**Sav**	**VS**	**RTR**
100	205.76	1.5	322.40	218.46	1.4	360.82
200	281.61	1.8	480.95	296.87	1.6	132.02
300	376.95	2.3	283.63	398.31	2.0	-96.54

Table 4.9: Results for the CEC with variations in TWw. The values of Sav, VS and RTR in each column are averages over all observations where TWw is equal to the corresponding value in row 1

Results for variations in the percentage of customers with LTL loads (%LD) is

given in Table 4.10. As %LD is increased, the distance savings increased as well. An increase in the number of customers with small demands allows the construction of routes of greater efficiency, without violating the capacity constraint of a vehicle. Similarly, an increase in %LD allows better loads to be created, which results in the higher IAS values. We present results for %LD only in the case $N = 100$, since runs for different values of N exhibited the same trend.

%LD	**80**			**95**		
N	**Sav**	**VS**	**IAU**	**Sav**	**VS**	**IAU**
100	204.92	1.4	5.00	219.31	1.4	6.39

Table 4.10: Results for the CEC with variations in %LD. The values of Sav, VS and RTR in each column are averages over all observations where %LD is equal to the corresponding value in row 1

In summary, entry-point collaboration reduced route distance and increased asset utilization significantly. Route time savings were not as high, except in a few cases.

4.3 Quadtree Search: Location of Transshipment Facilities

The preceding section discussed the results for entry point collaboration, which corresponded to steps 1 - 4 in Section 3.9. Now we proceed to analyze the performance of our quadtree search algorithm, using different heuristic evaluation functions. These results are used to choose the best heuristic function to locate transshipment facilities. The quadtree search method had an average execution time of only a few seconds on smaller problems. The average running time on the largest instance of 4800 customers was 40 seconds.

As mentioned in Section 3.5.1, we took a convex combination of terms in the evaluation function. The number of such terms was limited to two, so that tuning the quadtree search algorithm would be manageable in practice. In this section, we evaluate four potential heuristic functions. They are, with $\alpha_1 + \alpha_2 = 1$ in each case:

H1 $\alpha_1 \dfrac{C_{diff}}{K} + \alpha_2 \dfrac{A_{min}}{A_{quad}}$

H2 $0.5\alpha_1 \left(\dfrac{C_{diff}}{K} + \dfrac{n_{quad}}{\sum_{k=1}^{K} N_k} \right) + \alpha_2 \frac{A_{min}}{A_{quad}}$

H3 $\alpha_1 \dfrac{C_{diff}}{K} + \alpha_2 \left(\dfrac{n_{quad}}{A_{quad}} \times \dfrac{A_{min}}{\sum_{k=1}^{K} N_k} \right)$

H4 $\alpha_1 \dfrac{\sum_{k \in C_{quad}} \left[\delta_k \dfrac{n_k}{n_{k+1}} + (1 - \delta_k) \dfrac{n_{k+1}}{n_k} \right]}{C_{diff}} + \alpha_2 \dfrac{n_{quad}}{A_{quad}}$

$$\delta_k = \begin{cases} 1, & \text{if } n_k \geq n_{k+1}; \\ 0, & \text{otherwise.} \end{cases}$$

The first term in H1 represents the ratio of the number of different carriers (C_{diff}) in the quadrant to the total number of carriers in the problem (K). The second term is the ratio of the minimum allowable area (A_{min}) to the area of the quadrant (A_{quad}).

H1 tries to score the competing objectives that the clusters have customers from different carriers, and that those customers be spaced closely. The first objective recognizes that a cluster with customers from different carriers will lend itself to collaborative routes with higher probability. The second term captures the fact

that clusters with small area will likely result in higher customer density, and this will lead to better collaborative routes because of the close packing.

Our computational tests, however, showed that H1 yields clusters with poor point density, as the first term of the evaluation function does not encourage clusters with a sufficiently large number of points. To overcome this issue, we added a new term to H2, which is the number of points in the cluster (n_{quad}) divided by the total number of customers in the problem ($\sum_{k=1}^{K} N_k$). This term encourages clusters with a greater number of customers. Note that equal weights are asigned to the first term of H1 and to the new term, to form the aggregated first term of H2.

H3 also differs from H1 in trying to promote high density clusters. It achieves this by having a normalized density term (i.e. second term), given by the ratio of $\frac{n_{quad}}{\sum_{k=1}^{K} N_k}$ and $\frac{A_{quad}}{A_{min}}$. Observe that this normalized density will have a value of one when all customers are contained with in a cluster of area A_{min}.

The final evaluation function H4, has its second term identical to the second term of H1. However, H4 uses a different first term than H1 to promote the collaborative nature of the cluster. As we have explained before, two aspects of a cluster make good collaborative routes more probable. The first relates to cluster having customers from different carriers. The second is the high point density in clusters. Both these factors, however, overlook the situation where a cluster can have a large imbalance in the mixture of customers from different carriers.

For example, a problem with 4 carriers and a cluster with 20 customers could have a customer from each of the first three carriers and 17 customers from carrier 4. A cluster of this type is unlikely to promote collaborative routes. Therefore, term one of H4 tries to balance the number of customers from different carriers, by summing the pairwise ratio of the numbers of customers from different carriers. The parameter δ_k ensures that the ratio always contains a smaller number in the

numerator.

Each of the preceding evaluation functions has different competing objectives. To allow fair evaluation of the heuristic functions, common evaluation criteria were used. The first evaluation criterion (CTN1) is the customer point ratio, which is a ratio of the number of customers in the quadrant to the total number of customers in the problem. The second evaluation criterion (CTN2) is the number of different carrier points in the cluster divided by the number of carriers in the problem. Pairwise ratio between the number of customers from different carriers is given by the third (CTN3). The final statistic (CTN4) relates to the area ratio, which is given by $\frac{A_{quad}}{A_{tot}}$, where A_{tot} is the area of the city or the initial bounding rectangle.

In this section, we randomly select a VRPTW created for entry-point collaboration from each corner of the city. This implies that problems handled by the quadtree search have four times the number of points as the VRPTWs used in the collaborative-entry case. The number of customers for collaborative routing is N_c, where $N_c = 4N$ (i.e. l_2 from Step 3 of Section 3.9 equals 4). Figure 4.2 shows the variation of the different criteria with α_1 for $N_c = 400$. We do not show results for CTN2 as all findings indicate this to attain a best value of unity, which means that it does not discriminate between good and bad clusters.

Figs 4.2(a) and (b) show the variation of CTN1 with α_1. H3 dominates the other heuristics for most values of α_1, while H2 closely follows and actually does better for high values of α_1. H2 performs best in CTN3 as shown in Figs 4.2(c) and (d). H4 performs the worst in CTN3, but H2 and H3 have good average performance. Comparing the graphs for CTN1 and CTN3, it can be seen that H3 and H2 produce high density clusters, while H1 closely follows. The poor performance of H4 in CTN1 and 3, can be attributed to finding clusters with a balanced mixtures of customers [Figs 4.2(e) and (f)]. The CTN4 results indicate that the average performance of H4 is the best for $\alpha_1 \geq 0.6$.

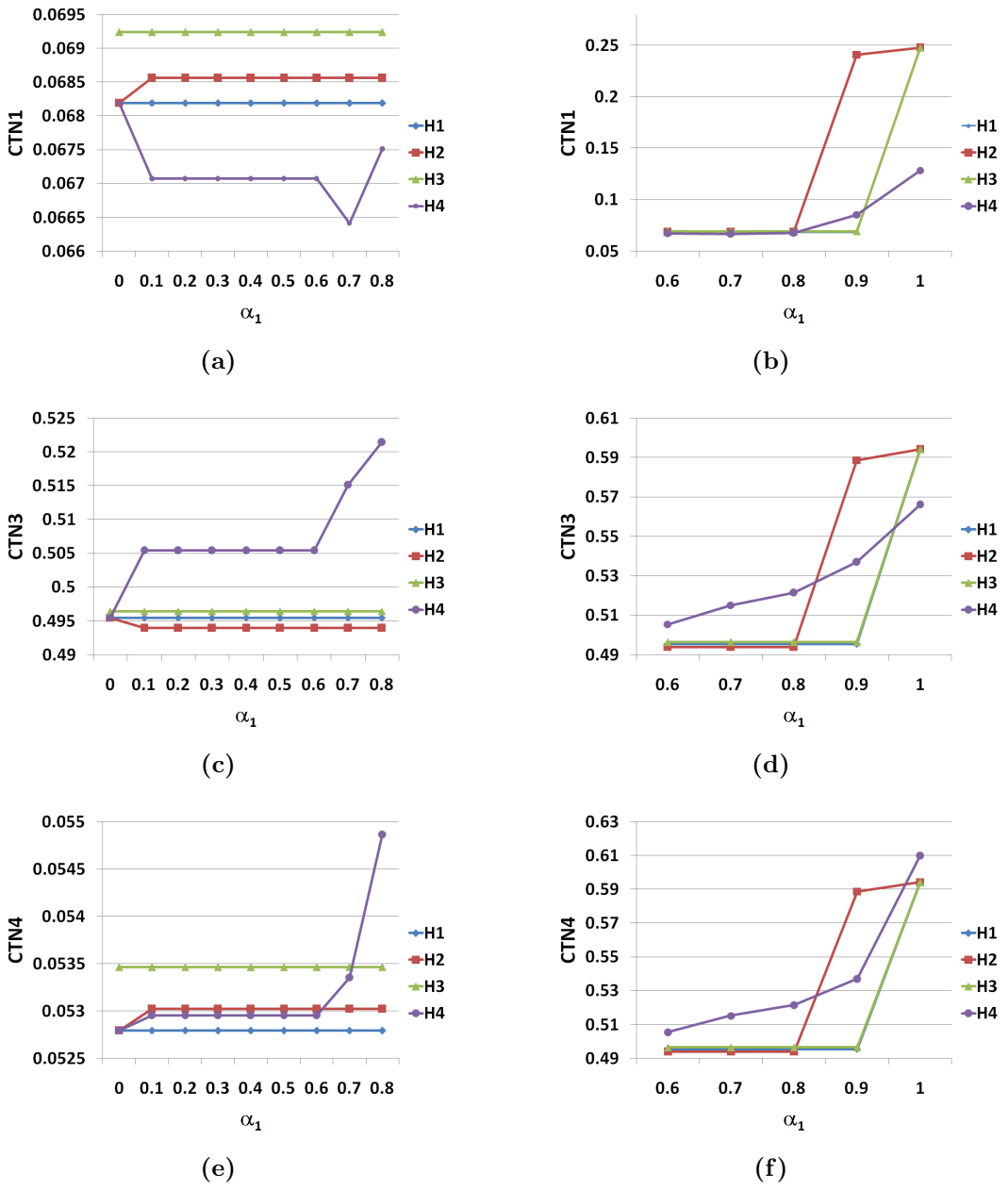


Figure 4.2: Variation of CTN1, CTN3 and CTN4 with α_1 for $N_c = 400$

The results for $N_c = 400$ indicate that clusters with high density, and clusters containing customers from different carriers, do not necessarily contain a *balanced* mixture of customers from different carriers. Our results for $N_c = 1200$ (Figure 4.4), also show this to be the case. For larger problems, it seems that H4 not only performs best for CTN4 but also for CTN3. The rankings of the heuristics for

criteria CTN1 and CTN3 [Figs 4.3(a)-(d)] imply that H4 exhibits good performance in terms of cluster density as well as for CTN4. Our computational test for $N_c = 800$ also showed the same trend. The poor performance of H4 under CTN1 and CTN3 for $N_c = 400$ is probably because of the sparse distribution of customers here, relative to larger problems.

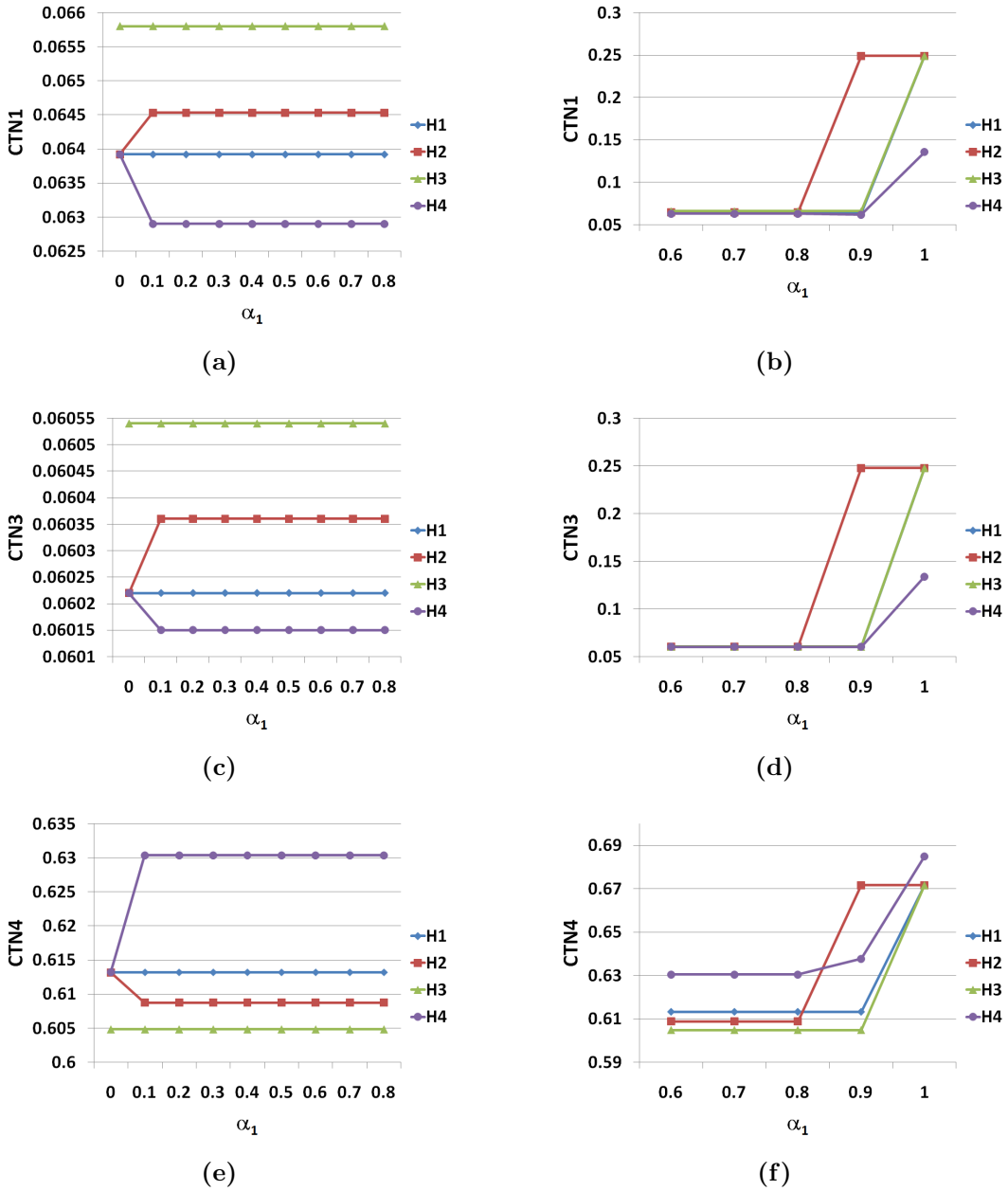


Figure 4.3: Variation of CTN1, CTN3 and CTN4 with α_1 for $N_c = 1200$

The next section will use the results from the greedy local search algorithm to further experiment with the quadtree search.

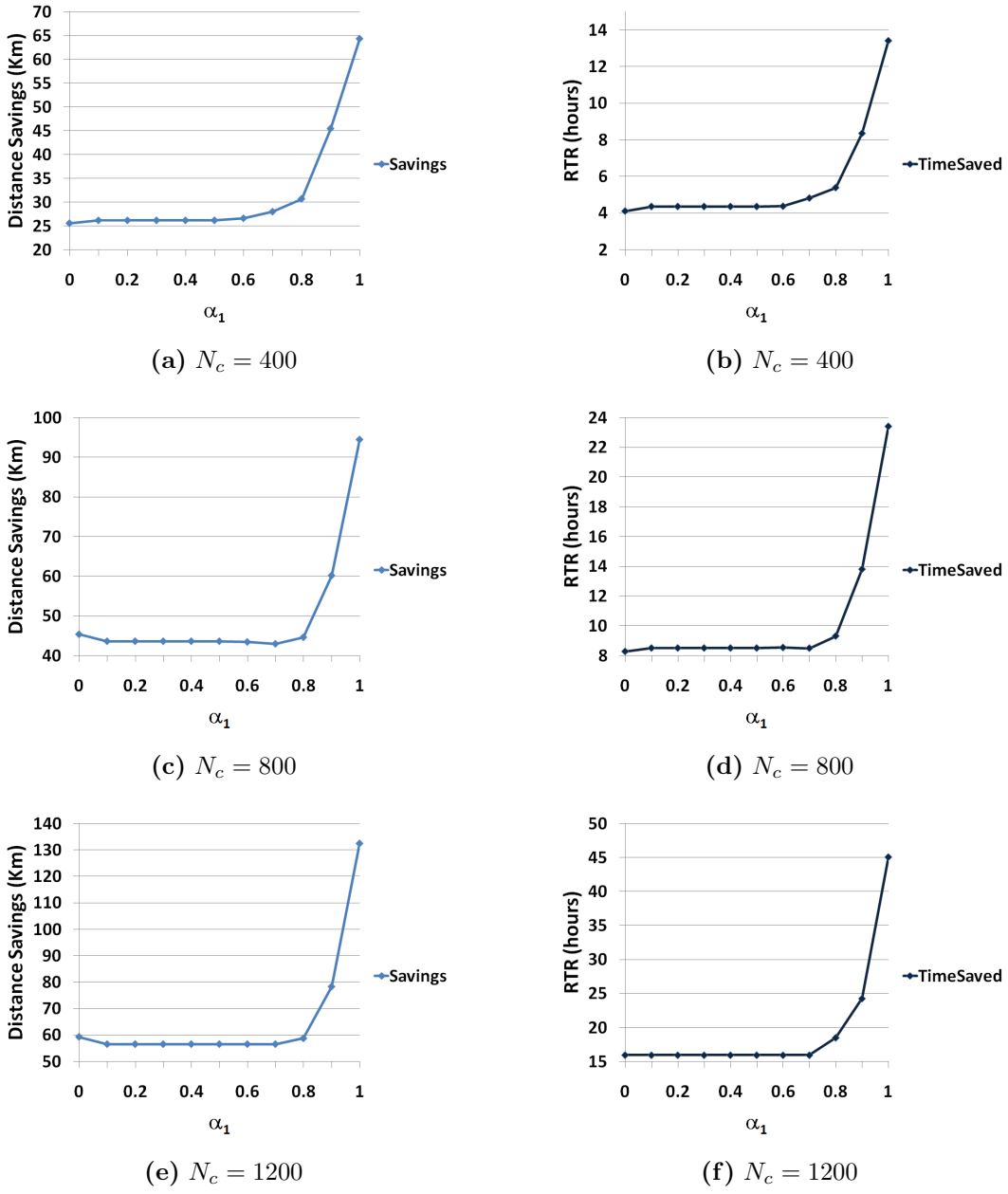
4.4 Greedy Local Search: Collaborative Routing

The second stage of collaboration, referred to as collaborative routing, involves the transfer of goods at transshipment facilities. The previous section presented some results for the quadtree search algorithm which was used to locate these facilities. Our greedy local search uses the routes from entry point collaboration and the location of potential transshipment facilities to construct collaborative routes. The distance savings and time savings from collaborative routing are presented in this section. The distance savings (Sav) are given in kilometers, while the time savings (RTR) are given in hours.

We selected H4 as the best heuristic using scores from predefined criteria in the previous section. However, no results were given regarding the variation with α_1 of the actual distance and time saved as a result of collaborative routing.

After solving the collaborative routing problem, results of both time and distance saved are shown as a function of α_1 in Figure 4.4. $\alpha_1 = 1$, results in the highest savings. This corroborates our assumption in the previous section that a cluster with a balanced mixture of customers from different carriers leads to better collaborative routes. From the average measures, we can also claim that more collaborative opportunities lead to both increased distance savings and time savings. The result for distance savings was expected since an "improving move" is defined using those savings. The substantial savings in time is an added bonus, and leads to reduction in congestion, as trucks spend less time in the city. The time savings refers to RTR, which is the total route time reduction.

There is still one more aspect of the quadtree search that we have not analyzed.



(a) $N_c = 400$ **(b)** $N_c = 400$

(c) $N_c = 800$ **(d)** $N_c = 800$

(e) $N_c = 1200$ **(f)** $N_c = 1200$

Figure 4.4: Variation of time and distance saved with α_1

This decision relates to the actual location of transshipment points. Once clusters are formed, the search has to use the information from customers present in the cluster to locate a transshipment point. We propose two methods for this: the demand weighted centroid (DW-Centroid) method and the time-slack weighted centroid method (TSW-Centroid).

Given a cluster p, let C_p represent the set of customers in the cluster. Following the notation of Section 3.2, let x_j, y_j, q_j and $sl_j = l_j - b_j$ represent the coordinates, demand and time slack of customer $j \in C_p$ respectively. We define the coordinates of the transshipment point $\left(x_p^{dw},\ y_p^{dw}\right)$ using the demand weighted centroid of a cluster as

$$x_p^{dw} = \frac{\sum_{c \in C_p} q_j x_j}{\sum_{c \in C_p} q_j} \tag{4.1}$$

$$y_p^{dw} = \frac{\sum_{c \in C_p} q_j y_j}{\sum_{c \in C_p} q_j} \tag{4.2}$$

Similarly, the coordinates of the TSW-Centroid can be derived by substituting q_j with sl_j in Eqns 4.1 and 4.2. Note that using DW-centroid results in the transshipment points located closer to customers with higher demand. Table 4.11 gives the average distance and time saved for the preceding two cases. The results are mixed, but if the distance and time saved are averaged over all values of N_c, TSW-Centroid is marginally better. However, our analysis showed that the average load transferred using DW-Centroid is about 8% greater than if TSW-Centroid is used. The transfer of larger loads results in higher asset utilization on the route, and often a greater reduction in dead-head miles.

In addition carriers would want to collaborate with other carriers on large loads, as this is more profitable, and better justifies the work at the transshipment facilities. We thus employ the results from DW-Centroid to locate those transshipment points.

To investigate the benefits of collaborative routing in more detail, we analyze the results with $\alpha_1 = 1.0$. Tables 4.12 and 4.13 present the variation with RType of distance and of time saved from collaboration, respectively. The results show that

	DW-Centroid		TSW-Centroid	
N_c	**Sav**	**RTR**	**Sav**	**RTR**
400	64.30	13.52	50.03	8.18
800	94.44	23.41	128.41	44.84
1200	132.48	45.05	135.08	44.00

Table 4.11: Results of average distance (Sav) and time (RTR) saved for DW-Centroid and TSW-Centroid. Sav and RTR are reported in kms and hrs respectively

the savings increase as Rtype changes from 1 to 2, and then decrease again when RType equals 3. This variation is due to a tradeoff between geographical advantage (Section 3.4) and route overlap.

For $RType = 1$, the routes are separated evenly and they all have the same geographical advantage from their entry point. When $RType = 2$, the routes from east and west (i.e. NE and NW, and SE and SW) overlap more, while the routes from the north and south (i.e. NE and SE, and NW and SW) overlap less. An increase in overlap leads to better collaborative routes, even though the east-west geographical advantage has diminished. However, when RType changes to 3, the savings drop, because the increase in overlap does not counter the effect of the decrease in geographical advantage. In addition, the overlap between north-south routes is much less, but collaboration between these routes, if it should take place, is certainly more beneficial now.

RType	**1**			**2**			**3**		
N	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**
400	66.20	57.84	74.06	66.31	48.73	82.96	60.78	49.59	70.96
800	86.36	74.90	132.95	108.09	76.90	132.95	82.43	65.92	100.89
1200	106.93	87.18	131.40	170.07	131.42	176.12	134.52	115.44	145.65

Table 4.12: Mean and quartile results for distance saved (kms) with variations in Rtype

Therefore, contrary to the results from entry-point collaboration, asymmetries in

the geometry can *help* collaborative routing. However, this is not to say that more asymmetry is always better, as there are tradeoffs as explained above. Though tradeoffs exist, results for the minimum average distance and time with RType equals 3, and N_c equal to $400, 800$ and 1200 are respectively 60.78, 82.43 and 106.93 kms, and 11.88, 19.23 and 41.50 hrs (Tables 4.12 and 4.13). These values indicate that collaboration is indeed beneficial, in both symmetric and asymmetric rectangular geometries.

RType	**1**			**2**			**3**		
N	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**
400	13.50	9.00	20.04	14.47	9.97	20.50	11.88	7.76	15.56
800	24.46	16.82	28.94	23.15	10.33	37.71	19.23	14.04	25.27
1200	47.06	39.49	54.65	51.71	31.70	61.87	41.50	31.96	45.69

Table 4.13: Mean and quartile results for time saved (hrs) with variations in Rtype

Analysis of the quartile results shows that collaborative routing leads to substantial savings in time, and distance savings as well. For $N_c = 800$ and $RType = 1$, Table 4.13 shows that route time is decreased by more than 16.82 hours for 75% of the observations, and 28.94 hours are saved on 25% of the samples. The distance saved is lower in comparison to the CEC (Table 4.7), and for the preceding setting, is more than 74.90 kms for 75% of the observations, and exceeds 132.95 kms for 25% of the cases (Table 4.12). Variations with $\%LD$ are not reported as they are similar to entry-point collaboration.

In collaborative routing, distance and time savings decrease (as expected) as a function of $\%TW$ (Tables 4.14 and 4.18). However, the variation of savings with TWw is the exact opposite: The results are counter-intuitive, as tighter time windows lead to *greater* time and distance savings than the case where the time windows are wider.

$\%TW$	**75**			**100**		
N	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**
400	73.72	59.64	70.05	55.14	46.78	62.50
800	101.28	73.07	116.95	83.29	66.47	100.23
1200	152.26	102.14	174.03	122.08	124.14	134.71

Table 4.14: Mean and quartile results for distance savings (kms) with $\%TW$.

$\%TW$	**75**			**100**		
N	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**
400	15.12	10.65	19.58	11.45	5.01	14.49
800	21.43	12.99	34.14	23.13	19.01	31.27
1200	48.26	34.68	48.91	45.25	29.63	61.60

Table 4.15: Mean and quartile results for time saved (hrs) with $\%TW$.

Although the number of collaborative opportunities will drop as time windows get tighter, the important effect is on the construction of VRPTW routes. Tight time windows lead to inefficient routes in terms of distance and contain only a few customers. Routes whose customers are a large distance from the depot tend to be less efficient. When carriers then exchange loads at a transshipment point the average savings from a collaborative opportunity increases substantially due to the initial inefficient routes. From a route-time perspective, tight time windows lead to excessive wait times as well. The trends observed in tables 4.16 and 4.17 are due to these reasons.

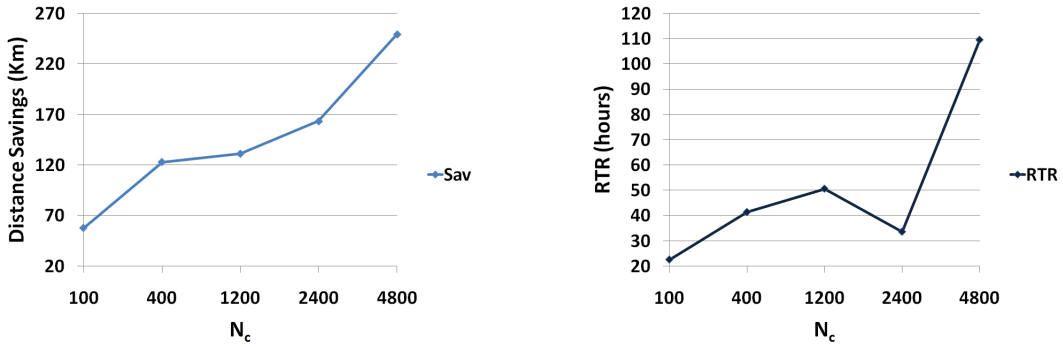
TWw	**2**			**5**		
N	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**
400	68.34	56.45	78.43	60.52	45.83	69.82
800	115.64	100.89	122.83	68.92	64.48	75.38
1200	168.12	131.52	176.01	106.22	93.18	120.93

Table 4.16: Mean and quartile results for distance saved (kms) with TWw.

TWw	2			5		
N	μ	**Q1**	**Q3**	μ	**Q1**	**Q3**
400	15.69	7.57	22.27	10.88	7.63	14.49
800	32.32	26.94	39.01	12.24	5.70	16.82
1200	51.77	33.39	62.49	41.74	30.52	48.19

Table 4.17: Mean and quartile results for time saved (hrs) with TWw.

Next, we present results for the case $RType = 1$, $\%TW = 75$, $TWw = 2$, $\%LD = 95$ and $N \in \{400, 800, 1200, 4800\}$. Figure 4.5 clearly shows that distance savings and route time reduction increase with N_c.



(a) **(b)**

Figure 4.5: Variation of time and distance saved with N_c when $RType = 1$, $\%TW = 75$, $TWw = 2$ and $\%LD = 95$

Table 4.18 contains the average number of transshipment facilities used to enable construction of collaborative routes. The number of facilities employed increases as the size of the problem increases, but our analysis of more granular data showed that not all transshipment sites selected can be profitably utilized. Some of these sites are used only for a few transfers and therefore the associated savings are low. In practice, this raises an important question of when a selected transshipment point should be actually be considered. We believe this depends on who owns the transshipment facility, and who takes the decisions regarding which carriers collaborate. These are challenges faced by any new proposal, and answers to these questions

can only be obtained from carriers and the government authorities themselves.

N_c	400	800	1200	2400	4800
ANTF	3.33	4.33	5.00	6.00	8.67

Table 4.18: Average number of transshipment facilities (ANTF) used for different values values of N_c

Finally, the average running time for greedy local search was 76.72, 210.68 and 580.09 seconds for N_c equals 400, 800 and 1200, respectively. Problems of size N_c equals 2400 and 4800 resulted in running times of 2141.9 and 3765.63 seconds. By restricting the size of the transferred customer sequence in each move operator (Section 3.8) to a maximum of four, we were able to reduce these running times by nearly half, with marginal reductions in savings.

Chapter 5

Summary and Conclusions

This chapter presents the summary of the work done and concluding remarks in Sections 5.1 and 5.2. Future research extensions are provided in Section 5.3.

5.1 Summary

The problem of collaboration between *LTL* carriers in an urban region is addressed in this thesis. To our knowledge, no literature exists on this problem. Therefore, we use simple examples to explain the benefits of carrier collaboration. These benefits include reduction in *dead-head miles* and in *extra miles*, an increase in *carrier revenue* and leads to both reduced *congestion* and *pollution*. From these benefits, we identify LTL trucking companies and the municipal government as potential beneficiaries of our work.

To solve the above problem, our main contribution was a two stage *collaborative framework*, which carriers can use to benefit from collaboration. These stages are:

1. *Collaborative entry*: Transfer of goods between trucks at logistics platforms located at the entry to the urban region.

2. *Collaborative routing*: Transfer of goods between trucks during local delivery at pre-specified transshipment sites.

The first stage of collaboration involves the solution of VRPTWs. We mathematically define the Collaborative Vehicle Routing Problem with Time Windows (COL-VRPTW), which corresponds to the second stage of collaboration. Two proofs, which show that collaboration leads to positive distance savings, are given for cases where carriers collaborate within a square. Then, graphical analysis of simple cases is used to show that the tradeoffs in collaboration can be non-intuitive. These results highlight that an algorithmic or mathematical study of carrier collaboration is warranted.

To solve the two stages described above, we provide a novel three-phase heuristic. Phase one uses an *integrated* tabu search (TS) or a guided local search (GLS) method to solve the VRPTWs encountered in entry-point collaboration. A randomized diversification strategy, motivated by satisfiability solvers, is employed to improve the performance of the TS used in Debacker and Furnon [3]. The GLS that we utilize is the same as in DeBacker et al. [4]. GLS is of course a local search method by itself. Our approach to that algorithm is an integrated one, whereby a CP engine aids the given local search method by acting as a "rule-checker." Similarly, our approach to the TS algorithm is also integrated.

The second phase of our algorithmic framework employs an *adaptive quadtree search* to create clusters of customers, given a COL-VRPTW instance. Once clusters are defined, based on a heuristic evaluation function, the preceding method utilizes information such as the customer demand and time windows to locate a transshipment point. By associating a transshipment point with a *cluster* of customers, we inherently apply a radius function to limit the set of customers considered during collaborative routing. Unlike this methodology, other VRP methods

use a radius parameter during local improvement. Complexity proofs for our implementation of the quadtree search are also given. We show that the quadtree depth is bounded by $\frac{3}{2} + \min[\log_2(\frac{l}{d}), 1 + \log_4(\frac{A_{init}}{A_{min}})]$, and that it has $O((d+1)n_{max})$ nodes and can be constructed in $O((d+1)n)$ time, where $n_{max} = \lceil \frac{n}{n_{min}} \rceil$. Definitions of the terms used in the preceding expression can be found in Appendix 5.3.

Our final algorithm is an integrated greedy local search method. Here we use the clusters and transshipment points from the quadtree search algorithm, together with routes from the first phase, to search for collaborative routing opportunities during local delivery. Three new transhipment-specific move operators for neighborhood definition were created. These move operators transfer a sequence of customers from one route to another, by exchanging goods at a transshipment point. The neighborhood of each move operator is exclusive.

In a different sense than before, the preceding algorithm is also an "integrated method," because an optimization model is used to group feasible moves encountered during neighborhood search. At each iteration, the optimization model returns a "greedy set" of moves, all of which will be combined and then used to transition from the current solution to a new one. By doing this, we were able to more efficiently utilize the information generated during neighborhood search. The optimization model to combine moves was solved using the commercial solver, ILOG CPLEX 11.

Finally, we perform extensive computational tests by solving over 10 000 VRPTWs. For our modified tabu search, we benchmark its performance on the VRPTW data set from Solomon [41]. For our three-phase heuristic, applied to carrier collaboration, we report results on random data sets that we created. Details of data-set creation are given in Appendix 5.3. The test city for our computations was Toronto. We also provide comprehensive evaluation of the results, and give managerial insights as to why our framework should be employed. A summary of the results and

conclusions drawn are given in the next section.

5.2 Conclusions

Our main contribution is a framework for LTL carrier collaboration and an accompanying solution methodology. Solution of this problem involved the definition of a new VRP variant which we term the "collaborative vehicle routing problem". We present geometric proofs in a square, for that problem under certain restrictions, which show that collaboration will lead to distance savings, as long as there is an overlap between collaborating routes.

For problems of realistic size, we define an integrated three-phase heuristic. The performance of our modified tabu search used in the first phase was tested on the data sets in Solomon [41]. When compared with other TS methods that minimize distance, the results showed that our modified TS was better than that in Tan et al. [44] for all data sets, and better than DeBacker and Furnon [3] for five out of six data sets.

Using TS and GLS on the random data sets that we created, we evaluated the impact of parameters such as city shape, percentage of customers with time windows, tightness of those time windows, and percentage of customers with LTL loads. Our results indicated that the distance savings from stage-one collaboration diminishes as the region deviated from a square. This result has important practical implications, as it suggests that not all cities are suitable for entry-point collaboration.

Further parameter analysis of entry-point collaboration showed that distance savings *decline* when there is more tightness in the time windows, or for a smaller percentage of LTL loads. The preceding observations are due to a reduction in

collaborative opportunities. A non-intuitive variation of distance saved with the percentage of customers having time windows was also observed. We argue that this variation occurs because increasing time windows leads to more restricted problems, which may benefit to a greater extent from collaboration.

Following that, we performed tests on our adaptive quadtree search to choose the best heuristic evaluation function. Computational tests showed that the best criterion for good collaborative clusters was a balanced mixture of customers from different carriers. Our parameter study on the evaluation function supported the preceding statement.

Once clusters were created, we tested two methods to locate a transshipment site. The first used a demand weighted centroid, while the second used a time-slack weighted centroid. Though the latter technique yielded slightly greater average savings, the demand weighted centroid resulted in the transfer of larger loads. From a practical standpoint, carriers would prefer transferring larger loads, since that implies higher asset utilization. Therefore, we chose the demand weighted centroid to locate facilities. Finally, we used an integrated greedy local search method to construct collaborative routes.

Next, we briefly summarize the benefits from collaboration and tie them to the three main goals of carrier collaboration: increase in carrier revenue, reduction in congestion, and reduction in pollution.

To begin, Table 5.1 shows the various percentage benefits. The first two columns give the percentage reduction in distance and time, respectively, for collaboration at the entry point to the city (CEC). Columns three and four contain the same two statistics for collaborative routing (CR). The last two columns relate to the combined, overall framework. Those findings correspond to the case $RType = 1$, $\%TW = 75$, $TWw = 2$ and $\%LD = 95$, the combination of parameters most likely

			CEC		CR		Overall	
n	N	N_c	$\%Sav$	$\%RTR$	$\%Sav$	$\%RTR$	$\%Sav$	$\%RTR$
25	100	400	12.37	2.06	3.58	15.44	15.51	17.18
50	200	800	8.96	2.27	3.90	13.24	12.51	15.21
75	300	1200	9.78	3.99	2.91	31.89	12.40	34.61
150	600	2400	10.31	-0.84	1.93	3.58	12.03	2.77
300	1200	4800	6.20	1.85	1.49	7.42	7.60	9.13

Table 5.1: Overall percentage reductions in route distance and route time when $RType = 1$, $\%TW = 75$, $TWw = 2$ and $\%LD = 95$

to occur in practice.

Our results show that distance savings from entry point collaboration are between 6.2 and 12.3%, depending on the number of customers. Such a decrease in route distance will also lead to substantial cost reductions, which will in turn provide an incentive for carriers to participate in collaboration. The reduction in route distance due to collaborative routing is between 1.4 and 3.9%. Though this is low compared to the collaborative entry case, as mentioned in Section 2.1.2, the cost of inter-city routing is $435 billion. Therefore, even a small percentage savings in distance can lead to considerable savings in *real* cost. With this in mind, the overall route distance savings from our proposed collaborative framework is very encouraging. The savings percentages are approximately between 3 and 15.5%. We believe that if savings in this range can be attained annually, carriers will be willing to participate in collaborative efforts.

Results for percentage route time reduction by collaborating at the logistics facilities at the entry point give savings between -0.9 and 4%. Though the reduction in time is not very high for this stage of collaboration, we reasoned in section 4.2, using our first and third quartile results, that the route time reductions were still substantial in most cases. In contrast, the savings in time from the collaborative

routing are quite high. For the values of n that we studied, collaborative routing resulted in route time reductions between 2.7 and 34.5%. Therefore, as a whole, the collaborative framework leads to significant savings in distance and time. Savings in time will permit the vehicle to return to the depot early, which may allow the utilization of that vehicle for either a second local delivery or for other hire purposes.

Results from Sections 4.2 and 4.4 show that average asset utilization can also be increased by about 4% and 8% for each stage of collaboration. This increase makes each vehicle more profitable, and leads to less dead-head miles as well.

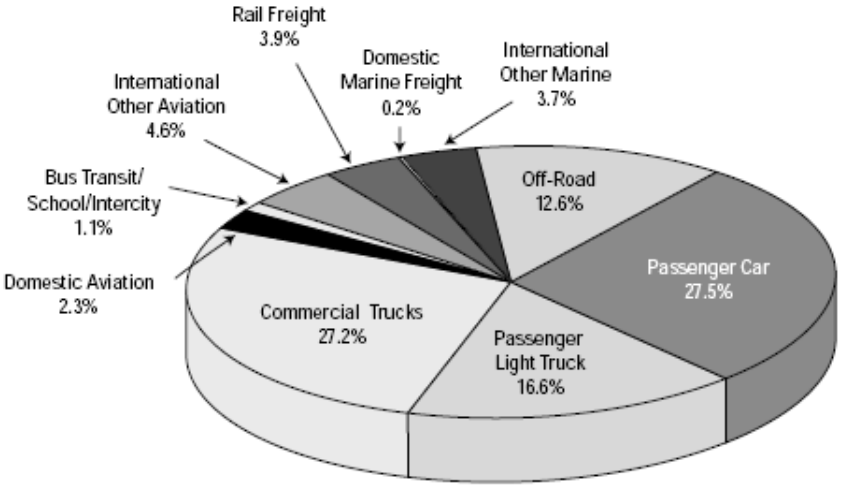


Figure 5.1: Green house gas emissions. Source: Transportation and climate change: Options for action. National climate change program, transportation table, November, 1999

Distance and time savings, in addition to providing financial and operational gains to carriers, also provide incentives for the city government to take our collaborative proposal into serious consideration. These incentives include route time reductions which lead to trucks spending less time doing local delivery. The reduced time that trucks spend in the city directly lessens congestion. It was estimated in Taylor [45] that the cost of congestion due to trucking in the city of Toronto and Peel regions was a shocking US$2 billion in the year 1987. Congestion in Toronto has increased considerably since then, and we expect the current cost of congestion

to be many times more. Therefore, reduction in congestion from our model has cost benefits as well.

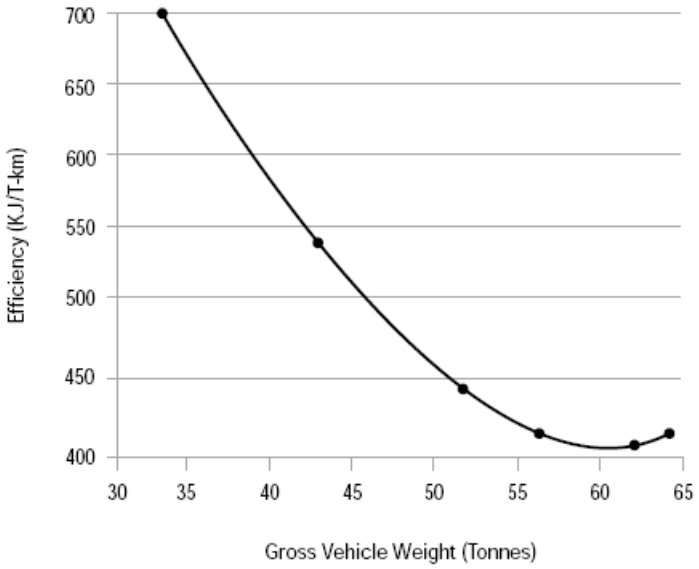


Figure 5.2: Truck weight and efficiency. Source: Trucks and air emissions, final report, Air pollution prevention directorate, Environmental protection service, Environment Canada, 2001

Trucking in the city has been a major contributor to pollution. Figure 5.1 shows commercial trucks to be the second biggest emitter of green house gases. Therefore, the reduction in route distance from collaboration of commercial trucks will have a direct impact on reducing pollution. Further, Fig 5.2 shows how the efficiency of a truck improves with weight. The increase in asset utilization from collaboration will therefore lead to enhanced trucking efficiency, which will lower pollution.

In this thesis, our framework for carrier collaboration was designed to provide a reduction in the carrier's routing costs and give environmental incentives for the local government to participate. Our results and evaluations confirm that LTL carrier collaboration in the city could indeed benefit these groups.

5.3 Research Extensions

We have analyzed carrier collaboration in urban regions, once the line haul delivery is complete. An open research question is to what extent line haul collaboration will help carriers and shippers? Work on shipper collaboration deals with this issue, but synergies between shipper and carrier collaboration have not yet been explored. For example, will combining both types of collaboration lead to a higher combined payoff, or individually higher payoffs? This has serious implications to government authorities who may try to implement such frameworks.

The presentation and evaluation of results in this thesis was done to prove that collaboration was beneficial. Our algorithmic contributions, such as the the modified tabu search and the integrated greedy local search, warrant more exploration in terms of performance testing and benchmarks.

Further, during the development phase of the modified tabu search, experiments on using predicates to reduce the neighborhood size and strategically intensify the search showed promise. This can be partially attributed to the highly efficient propagation algorithms for simple predicates in CP. That is a research direction, independent of this thesis, which is being pursued. In general, problems in collaborative logistics are mathematically complex, and therefore are rich in opportunities for developing good heuristic methods.

Finally, another unexplored avenue is the development of a holistic framework for collaboration which accounts for inventory replenishment issues. This may lead to great benefits as well.

Appendix A

Proof of Theorems

Definition .0.1. Quadtree: A quadtree Q is a rooted unbalanced tree. Each internal node has four children.

Theorem .0.2. *The maximum depth of a quadtree Q is bounded by* $\frac{3}{2}+\min[\log_2(\frac{l}{d}), 1+\log_4(\frac{A_{init}}{A_{min}})]$*, where A_{min} is the minimum allowable area of a quadrant and A_{init} is the initial area of the bounding rectangle.*

Proof. Q has two termination criteria based on minimum allowable quadrant area and minimum number of points within a quadrant

1. **Minimum Area**

   At depth i, A_i is the area of a quadrant

$$\Rightarrow A_i = \frac{A_{init}}{4^i}$$

$$\therefore \frac{A_{init}}{A_{min}} \geq 4^i$$

$$\Rightarrow i \leq \log_4(\frac{A_{init}}{A_{min}})$$

Since the depth of a tree is one more than the maximum depth of an internal node, the depth of the quadtree (d_Q) satisfies:

$$d_Q \leq 1 + \log_4(\frac{A_{init}}{A_{min}})$$

2. **Minimum Number of Points**

   Let there be a total of n points in the bounding rectangle. Let S contain all the subsets of points in the bounding rectangle such that each set $s \in S$ has $card(s) = n_{min}$, where n_{min} is the minimum number of points that must be present in a quadrant.

   Let us define:

$$Sep_{min} = \{\max(d) : d = \sqrt{[s(x_i^p) - s(x_k^q)]^2 + [s(y_i^p) - s(y_k^q)]^2}, s_i, s_k \in S \\ , i \neq k, x_i^p \in s_i, x_k^q \in s_k\}$$

   Let l_{init} be the length of the largest side of the bounding rectangle.

   $\Rightarrow$ At a depth i, the side of the current square corresponding to the largest side will have length $l_i = \frac{l_{init}}{2^i}$

   Within a quadrant the largest separation between two points is bounded by the diagonal distance, which at depth i is $diag_i = \frac{\sqrt{2}l_{init}}{2^i}$

The termination criteria imposes that

$$\frac{\sqrt{2}l_{init}}{2^i} \geq Sep_{min}$$

$$\Rightarrow \frac{\sqrt{2}l_{init}}{Sep_{min}} \geq 2^i$$

$$\Rightarrow i \leq \log_2(\frac{\sqrt{2}l_{init}}{Sep_{min}})$$

$$\Rightarrow i \leq \log_2(\frac{l_{init}}{Sep_{min}}) + \frac{1}{2}$$

Since the depth of a tree is one more than the maximum depth of an internal node, the depth of the quadtree (d_Q) satisfies:

$$d_Q \leq \log_2(\frac{l_{init}}{Sep_{min}}) + \frac{3}{2}$$

As one of the above bounds must hold strictly, the bound on the depth follows.

□

Theorem .0.3. *A Quadtree Q of depth d and storing n points, has $O((d+1)n_{max})$ nodes and can be constructed in $O((d+1)n)$ time, where $n_{max} = \lceil \frac{n}{n_{min}} \rceil$*

Proof. Each internal node has four children and the total number of nodes can be derived from the number of internal nodes. Therefore, it is sufficient to analyze the internal nodes alone. At a given depth, the internal nodes representing different quadrants are disjoint and cover the entire bounding rectangle. This implies that they collectively store at most n points. Since the minimum number of allowable points within a quadrant is n_{min}, the maximum number of nodes at a given depth

is $n_{max} = \lceil \frac{n}{n_{min}} \rceil$. From this, the bound on the number of nodes follows.

The main operation at every node is to assign points to different quadrants. The time consumed by this operation is linear in the number of nodes in the current square. Since the maximum number of points which can be associated with a square is n, the time bound follows. □

Appendix B

Data Set Creation

In this appendix, details regarding the method used to create data sets for the VRPTW will be described. These are subsequently used to create the COL-VRPTW data sets as explained in section 3.9. Recall that the test region is rectangular and represents a city. The corners of the rectangle are the points of entry into the city, where logistics platforms are located. The data sets need to contain information regarding the entry points or vehicle depots (i.e NW, NE, SE or SW entry point), starting and return times of vehicles, vehicle capacity, spatial distribution of customers and their time windows, service time at a customer and the service requirement there (i.e. quantity of goods to be delivered).

The user needs to provide information such as the rectangular regions length (L) and breadth (B), the number of customers, n, the vehicle capacity, C_v, the percentage of customers with small demand, p_1, and the percentage of customers with time windows, p_3.

The vehicle depot is chosen by generating a uniformly distributed random integer between 1 and 4 (i.e. NW = 1, NE = 2, SE = 3, SW = 4). Next the customer coordinates (x, y) are generated. The x coordinate of each customer is randomly selected from an equilikely integer random distribution between zero and the length of the rectangle. y is selected in a similar manner, but the range of the distribution

is between zero and the breadth of the rectangle. Once n coordinates have been generated, the spatial distribution of customers is fixed. The starting time of a vehicle is set to zero, without loss of generality.

The latest return time of a vehicle to the depot is set to be 13 hours (i.e 780 minutes). This restriction is imposed by the truck drivers maximum hours of operation (http://gazetteducanada.gc.ca). The service time, S_t at each customer is set to 20 minutes, based on the unloading rate of an average LTL load by two drivers. The average speed of a truck in an urban region is assumed to be 30 kilometers per hour. We use this speed to link time and distance.

The demand is obtained as follows. A uniform random number between 0 and 1 is generated for each customer. If the number generated is less than or equal to $p_1/100$ a random value from an uniformly distributed distribution over the range $[0.01C_v, 0.36C_v]$ is assigned. This percentage range is based on the fact that, LTL loads are around 500-15000 lbs. If the random number is greater than $p_1/100$ a random demand is assigned from an uniformly distributed random distribution within the range $[0.37C_v, 0.5C_v]$. The second set of higher demands are to account for the random nature of LTL shipments (Chapter 2.7.2, Page 10). However, the percentage of these high quantity requests is small.

Following this, we generate time windows for each customer. A uniform random number between 0 and 1 is generated for each customer. If the random number is greater than $p_3/100$, we assign a time window identical to that of the depot (i.e. the customer has no time window). If the random number is less than or equal to $p_3/100$ we generate a time window as explained below. The center of the time window for customer i is generated via a uniform random number between $[e_t + t_{0i}, l_0 - t_{i0} - S_i]$, where S_i is the service time and t_{0i} is the travel time from the depot to the customer.

Next we use a truncated normal distribution to generate the half width of the time window. A truncated normal distribution is a normal distribution which is bounded above and below. To define a truncated normal distribution $TN(\mu, \sigma^2, a, b)$, the following need to be specified: the mean, μ, standard deviation σ, the lower bound a, and the upper bound b. μ is equal to the time window center just generated, and a and b are $[e_t + t_{0i_j}$ and $l_0 - t_{i_j 0} - S_{i_j}]$ respectively. The standard deviation is used to vary the tightness of time windows. This concludes the procedure for generating a VRPTW data set.

References

[1] M. Aichlymayr. Dc mart: who manages inventory in a value chain? *Transportation and distribution*, 41(10):60–68, 2000. 5

[2] B.J. Angerhofer and M.C. Angelides. A model and a performance measurement system for collaborative supply chains. *Decision support systems*, 42(1):283–301, 2006. 5

[3] B. De Backer and V. Furnon. Meta-heuristics in constraint programming: Experiments with tabu search on vehicle routing problems. In *Second international conference on metaheuristics (MIC'97)*, Sophia Antipolic, France, 1997. 29, 59, 77, 78, 100, 102

[4] B. De Backer, V. Furnon, P. Kilby, P. Prosser, and P. Shaw. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of heuristics*, 6(4):501–523, 2000. 29, 31, 40, 100

[5] J. H. Bookinder and K. E. Reece. Vehicle routing considerations in distribution systems design. *European journal of operational research*, 37(2):204–213, 1988. 51

[6] O. Bräysy and M. Gendreau. Vehicle routing with time windows, part 2: Metaheuristics. *TOP*, 10(2):211–237, 2002. 26

[7] O. Bräysy and M. Gendreau. Vehicle routing with time windows, part 1: Route construction and local search algorithms. *Transportation science*, 39(1):104–118, 2005. 39

[8] O. Bräysy and M. Gendreau. Vehicle routing with time windows, part 2: Metaheuristics. *Transportation science*, 39(1):119–139, 2005. 26, 39

[9] Y. Caseau and F. Laburthe. Heuristics for large constrained vehicle routing problems. *Journal of heuristics*, 5(3):281–303, 1999. 40

[10] J.F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. *The vehicle routing problem*, volume 9 of *SIAM monographs on discrete mathematics and applications*, chapter 7, pages 157–193. SIAM, Philadelphia, 2001. 28, 29, 39

[11] T. G. Crainic, N. Ricciardi, and G. Storchi. The-day-before planning for advanced freight transportation systems in congested urban freight. Technical Report CRT-2005-19, Centre for research on transportation, University of Montreal, Montreal, Canada, 2005. 50

[12] T. G. Crainic, N. Ricciardi, and G. Storchi. Routing vehicles in two-level city logistics systems. 50th Canadian Operational Research Society Conference 2008, University of Laval, Quebec, May 2008. Canadian Operational Research Society. 50

[13] J. Desrosiers, G. Laporte, M. Suave, F. Soumis, and S. Taillefer. Vehicle routing with full loads. *Computers and operations research*, 15(3):219–226, 1988. 10

[14] O. Ergun, G. Kuyzu, and M. Savelsbergh. Reducing truckload transportation costs through collaboration. *Transportation Science*, 41(2):206–221, 2007. 11, 12

[15] O. Ergun, G. Kuyzu, and M. Savelsbergh. Shipper collaboration. *Computers and operations research*, 34(6):1551–1560, 2007. 11

[16] R.A. Finkel and J.L. Bentley. Quadtrees: A data stucture for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974. 15

[17] H. Gehring and J. Homberger. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of heuristics*, 8(3):251–276, 2002. 28, 29

[18] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and operations research*, 13(5):522–549, 1986. 26, 27

[19] F. Glover. Tabu search - part i. *Journal on computing*, 1:190–206, 1989. 27

[20] F. Glover. Tabu search - part ii. *Journal on computing*, 2:4–32, 1990. 27

[21] M. Gumus, J.H. Bookbinder, and E.M. Jewkes. Calculating the benefits of vendor managed inventory in a manufacturer-retailer system. Working paper, University of Waterloo, Waterloo, Canada, August 2006. 6

[22] P. Hansen. Technical report. 27

[23] P. Van Hentenryck and Laurent Michel. *Constraint-based local search*. MIT press, Cambridge, Massachusetts, 2005. 21

[24] J. N. Hooker. *Integrated methods for optimization*. Springer, New York, USA, 2007. 20

[25] P. Kilby, P. Prosser, and P. Shaw. A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. *Constraints*, 5(4):389–414, 2000. 40

[26] P. Kilby and P. Shaw. *Vehicle routing*, chapter 23, pages 801–836. Foundations of Artificial Intelligence. Elsevier, first edition, October 2006. 55, 57

[27] G.A.P. Kindervater and M.W.P. Savelsbergh. *Vehicle routing: Handling edge exchanges*, chapter 10, pages 337–360. Local search in combinatorial optimzation. Princeton University Press, Princeton, New Jersey, 2003. 59, 63

[28] J.C. Langley. 7 immutable laws of collaborative logistics. White paper, Nistevo, Minnesota, USA, July 2000. 6

[29] G. Laporte. What you should know about the vehicle routing problem. *Naval research logistics*, 54(8):811–819, 2007. 37

[30] G. Laporte, M. Gendreau, J-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300, 2000. 23

[31] K. Lynch. Collaborative logistics networks: Breaking traditional performance barriers for shippers and carriers. White paper, Nistevo, Minnesota, USA, 2001. 7

[32] S. Mancini, T. G. Crainic, G. Perboli, and R. Tadei. A satellite location study for the two-echelon vehicle routing problem. 50th Canadian Operational Research Society Conference 2008, Université Laval Québec, May 2008. Canadian Operations Research Society. 50

[33] D. Mester and O. Bräysy. Active guided evolution strategies for the large-scale vehicle routing problem with time windows. *Computers and operations research*, 32(6):1593–1614, 2005. 39

[34] S. Mitrović-Minić and G. Laporte. Pickup and delivery problem with time windows and transshipment. *INFOR*, 44(3):217–227, 2006. 37

[35] S. Robin and R.R. Levary. Vehicle routing via column generation. *European journal of operational research*, 21:65–76, 1985. 10

[36] Y. Rochat and É.D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics*, 1(1):147–167, 1995. 28, 39, 61

[37] D. Ronen. Dispatching continuous moves. *Journal of transportation management*, 16(2):25–37, 2005. 11

[38] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of knowledge representation*, chapter 4, pages 181–212. Foundations of Artificial Intelligence. Elsevier, Amsterdam, Netherlands, 2007. 17

[39] S. Russell and P. Norvig. *Artificial intelligence: A modern approach*, chapter 7. Prentice Hall, New Jersey, USA, 2nd edition, 2003. 59

[40] M. Savelsbergh and M. Sol. The general pick-up and delivery problem. *Transportation science*, 29(1):17–29, 1995. 10

[41] M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987. 3, 29, 31, 75, 101, 102

[42] J.L. Sutherland. Collaborative transportation management: A solution to the current transportation crisis. CVCR white paper 0602, Lehigh University, Pennsylvania, U.S.A, 2006. 7

[43] É.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997. 31, 39

[44] K. C. Tan, L. H. Lee, and K. Q. Zhu. Heuristic methods for vehicle routing problems with time windows. In *Proceedings of the 6th international symposium on artifical intelligence and mathematics*, Ft. Lauderdale, FL, 2000. 29, 77, 102

[45] G.W.R. Taylor. Trucks and air emissions. Technical report, Air Pollution Prevention Directorate, Environmental Protection Service, Environment Canada, September 2001. 1, 105

[46] P. Toth and D. Vigo, editors. *The vehicle routing problem*, volume 9 of *SIAM monographs on discrete mathematics and applications*. SIAM, Philadelphia, 2001. 37

[47] C. Voudouris and E. Tsang. Guided local search. *European journal of operational research*, 113(2):80–110, 1998. 26, 30

[48] C. Voudouris and E. Tsang. *Guided Local Search*, chapter 7, pages 185–218. International Series in Operations Research and Management Science. Kluwer Academic Publishers, 2002. 26, 30

[49] M. Waller, M.E. Johnson, and T. Davis. Vendor-managed inventory in the retail supply chain. *Journal of business logistics*, 20(1):183–203, 1999. 6

[50] R. Wilson. *CSCMP's annual state of logistics report*. CSCMP, Illinois, 2007. 1, 9