PROCUREMENT IN TRUCKLOAD TRANSPORTATION

A Thesis
Presented to
The Academic Faculty

by

Gültekin Kuyzu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2007

PROCUREMENT IN TRUCKLOAD TRANSPORTATION

To my family.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisors Dr. Martin Savelsbergh and Dr. Özlem Ergun. I was extremely fortunate to work with them. They provided me with the motivation, guidance, and support I needed to complete my Ph.D. studies. I would like to especially thank them for giving me the freedom to choose the topics I wanted to study. Their constant interest in my research reassured me that I was working on challenging and interesting problems.

I would like to thank Dr. Shabbir Ahmed, Dr. Alan Erera, and Dr. Jun-Sheng Li for agreeing to serve on my committee by sparing their valuable time and for providing helpful comments and suggestions.

I would like to thank my parents for everything they have done to help me succeed throughout my life. I left my parents' home at a relatively young age to study high school in a boarding school. Then, I came to the U.S. for my undergraduate studies, which was followed by my Ph.D. studies. My education was so important for my parents, and my whole family, that they sometimes held back their true feelings and the events in their lives thinking that they would distract my studies. I am sincerely grateful to my parents Ömer and Gülsen, my sister Dilşad, my brothers Zafer and Can, and all of the rest of my family for all of their sacrifices and their unconditional love and support.

I would like to especially thank my lovely wife, Melike, for all her love and companionship. It would have been so much more difficult to complete this dissertation without her presence. Being the person closest to me, she had to suffer through the ups and downs of my Ph.D. studies, but she never stopped bringing my life joy and happiness, which I needed the most.

I would like to thank my officemate Burak Karacık, and fellow Ph.D. students Andrei Prudius and Yetkin Íleri, with whom I had several stimulating and enjoyable conversations about research, technology, and life. I would like to thank my dear friends Buğra Gedik, Tarık Arıcı and Şeyhmus İnci for all the times we spent together. I would like to thank Ahmet Gürer, Engin Can, Fehmi Büyükdereli, Halil Karayağlı and Mustafa Başgün for getting me and my mind out of Georgia Tech every once in a while. I would like to thank Hüseyin Özsert, my close friend since high school, for keeping in touch with me even when we were in different continents. Finally, I would like to thank Muharrem Aktaş, Süleyman Çelik, Ramazan Başgut, Borga Deniz, Caner Kazancı, Galip Kıyaklı and Mustafa Yaşar Sır for being great friends throughout my undergraduate and graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

SUMMARY

In this thesis, we address a number of operational challenges encountered in two emerging types of practices in the procurement of truckload transportation services: collaboration and auctions. The main objective in these two types of procurement strategies is identifying and exploiting synergies between the lanes of a carrier's network. In shipper collaboration, we take the perspective of a shipper who collaborates with other shippers to seek synergy between his lanes and other participants' lanes. On the other hand, in procurement auctions, although we take the carriers' perspective in our work, a shipper tries to discover the carrier (or carriers) whose network has the most synergy with his lanes.

The first part of the thesis concerns the solution of optimization problems arising in collaborative transportation procurement networks where a group of shippers comes together and jointly negotiates with carriers for the procurement of transportation services. In the highly fragmented truckload transportation industry a substantial fraction of truck movements involves empty trucks, i.e., involves moves that reposition trucks. However, reducing the amount of truck repositioning is difficult, because the total repositioning movements of a carrier depends on the interactions between the shippers the carrier is serving. Through collaboration, shippers may be able to identify and submit sequences of continuous loaded movements to carriers, reducing the carriers' need for repositioning, and thus lowering the carriers' costs. A portion of the carriers' cost savings may be returned to the shippers in the form of lower prices. We discuss optimization technology that can be used to assist in the identification of repeatable, dedicated truckload continuous move tours with little truck repositioning. Timing considerations are critical to practical viability and are a key focus of our efforts. We demonstrate the effectiveness of the algorithms developed on various randomly generated instances as well as on instances derived from data obtained from a strategic sourcing consortium for a \$14 billion dollar sized US industry.

The second part of the thesis concerns the pricing of transportation services offered by the trucking companies (carriers). We look at the bid determination problem faced by carriers in transportation procurement auctions where a shipper requests quotes from the carriers and purchases the services of the lowest bidder. Determining the optimal bid values for a carrier poses several challenging research questions. The specific problem being studied is the bid valuation problem in the case where the carrier must place bids on multiple lanes simultaneously. For each lane, the carrier's bid must be high enough to make it profitable for the carrier but low enough to give the carrier a good chance of winning that lane. The difficulty is that the operating cost of the carrier depends on the set of lanes the carrier wins which is not known until the end of the auction. We model the problem as a stochastic optimization problem and propose a coordinate search algorithm for solving this problem. Then, we conduct a simulation study to demonstrate the positive impact of the approach on carrier profits.

In both problems considered in the thesis, the central issue is identifying and exploiting synergies between the lanes of a carrier's network.

CHAPTER I

INTRODUCTION

Cost-effective transportation of goods is critical to the success of many of today's manufacturing and retail companies. A significant portion typically involves truckload transportation, which is purchased from truckload carriers who charge their customers based on the distance between the pickup and delivery locations of the load and the number of truckloads to be transported. Intense competition has caused carriers to operate with low profit margins. As a result, carriers constantly seek ways to reduce operating costs. A substantial fraction of a truckload carrier's operating costs comes from the repositioning movements of its trucks (no revenue is generated as the truck moves empty, but costs are incurred). Consequently, minimizing repositioning movements of trucks is one of the primary objectives of dispatchers of truckload carriers. Unfortunately, this is easier said than done. Total operating cost of a carrier is a result of the carrier's overall interaction with its customers. The cost of serving a single lane depends not only on the distance between the origin and destination of that lane but also its relation to other lanes served by the carrier (since those impact the necessary repositioning movements). Therefore, identifying and exploiting synergies between the lanes of a carrier's network is key to reducing transportation related operating costs.

Recent developments in information and communication technologies are opening the way for new opportunities for improving the current business practices. Transportation procurement is no exception. Internet allows companies, both shippers and carriers, to communicate in a much faster and more cost effective way than they used to. Recently, transportation procurement using internet enabled logistics networks or exchanges has been on the increase. In this thesis, we try to address a number of operational challenges encountered in two emerging classes of logistics networks with two separate but closely related principles: collaboration and auctions. The common focal point in these two classes procurement practices is identifying and exploiting synergies between the lanes of a carrier's network. In collaborative logistics networks, e.g. run by Elogex, Nistevo, or Transplace, a shipper seeks synergy between his lanes and other participants' lanes. On the other hand, in procurement auctions, a shipper tries to discover the carrier (or carriers) whose network has the most synergy with his lanes.

The first part of the thesis concerns the solution of optimization problems arising in collaborative logistics networks where a group of shippers come together and jointly negotiate with carriers on a long-term contract basis for the procurement of transportation services. Collaboration, in a variety of forms, is a fairly recent business practice which is becoming increasingly popular. The ever increasing pressure to operate more efficiently leads companies to seek and adopt solutions beyond the boundaries of traditional business practices. Instead of focusing on internal business processes, collaboration focuses on system-wide performance and aims to take advantage of synergies with collaborative partners. In collaborative logistics networks, the members (shippers) try to identify dedicated, repeatable continuous move routes for which carriers tend to offer lower prices. Due to the size of the network and the constraints on the routes that can be formed, determining the best set of collaborative continuous move routes presents a challenging optimization problem.

We are the first to study the problem of determining the optimal set of collaborative continuous move routes. In this thesis, we introduce the resulting optimization problem, namely the Lane Covering Problem (LCP), and a number of its constrained variants and propose and analyze a variety of solution approaches. A key focus is on timing considerations since they are critical to practical viability. We propose a solution approach utilizing greedy construction heuristics and local improvement. We demonstrate the effectiveness of the proposed solution approaches on various randomly generated instances that mimic real-life supply chain structures, and on instances derived from real-life data.

The second part of the thesis concerns the pricing of transportation services offered by the truckload carriers participating in transportation procurement auctions where a shipper requests quotes from the carriers and purchases the services of the lowest bidder. Auctions in transportation procurement have been mostly analyzed from the shipper's perspective by researchers and practitioners alike. The carriers participating in transportation procurement auctions face significant challenges. Their bids must be high enough to make it profitable for the carrier but low enough to give the carrier a good chance of winning the lane(s) involved in the auction. The availability of effective tools and strategies to help the carriers determine bids in procurement auctions is limited. Development of such tools and strategies would help not only the carriers but also the shippers in terms of reduced transportation costs, improved service, and so forth.

The specific problem studied is the bid pricing problem in the case where the carrier must place bids in multiple independent single lane auctions simultaneously. The difficulty is that the operating cost of the carrier depends on the set of lanes the carrier wins which is not known until the end of the auction. We discuss carrier bidding models for a carrier considering the risk of losing lanes to another carrier. The emphasis is on incorporating the bidding behavior of the competitors in order to account for the uncertainty involved with the set of lanes the carrier ends up winning.

We are the first to study simultaneous transportation procurement auctions from a carrier's perspective. The existing literature on carrier bidding strategies has focused on either combinatorial or sequential auction settings. We assume that the lowest price quoted by the carrier's competitors is a random variable and formulate a stochastic optimization problem with the objective of maximizing the carrier's expected profit. We devise and implement an iterative optimization algorithm and demonstrate its efficiency and effectiveness as compared to general purpose optimization algorithms on randomly generated instances. We also show the positive impact of our algorithm on the carrier profits through a simulation study.

The thesis is organized as follows. Chapter 2 covers shipper collaboration and Lane Covering Problems. Section 2.3 discusses the solution of cardinality and length constrained variants of LCP which ignore timing considerations. Section 2.4 extends the discussion to the time constrained LCP which incorporates timing considerations into the problem.

Chapter 3 presents bidding algorithms for carriers participating in simultaneous transportation procurement auctions. Chapter 4 concludes the thesis with a summary of our findings and possible future directions.

CHAPTER II

SHIPPER COLLABORATION

2.1. Introduction

The growing interest in collaborative logistics is fueled by an ever increasing pressure on companies to operate more efficiently, the realization that suppliers, consumers, and even competitors, can be potential collaborative logistics partners, and the connectivity provided by the Internet.

In the trucking industry, shippers and carriers are continuously facing pressures to operate more efficiently. Traditionally shippers and carriers have focused their attention on controlling and reducing their own costs to increase profitability, i.e., improving those business processes that the organization controls independently. More recently, shippers and carriers have focused their attention on controlling and reducing system wide costs and sharing these cost savings to increase every one's profit. A system wide focus, e.g., a collaborative focus, opens up cost saving opportunities that are impossible to achieve with an internal company focus. A good example is asset repositioning. To execute shipments from different shippers a carrier often has to reposition its assets, i.e., trucks. Shippers have no insight in how the interaction between their various shipments affects a carrier's asset repositioning costs. However, shippers are implicitly charged for these repositioning costs. No single participant in the logistics system controls asset repositioning costs, so only through collaborative logistics initiatives can these costs be controlled and reduced.

Collaborative transportation networks, such as those managed by Nistevo [61] and Transplace [80], are examples of collaborative logistics initiatives focused on bringing together shippers and carriers to increase asset utilization and reduce logistics costs. Services offered range from identifying continuous moves, i.e. consecutive loaded moves with little or no repositioning in between, or collaborative continuous moves to identifying repeatable dedicated continuous move tours to consolidation of in- and outbound transportation.

As an example of the value of collaboration, consider a repeatable dedicated 2,500-mile continuous move tour set up for two of the members of the Nistevo network (See Figure 2.1). The tour visits distribution centers, production facilities, and retail outlets. The tour has resulted in a 19% savings for both shippers (over the costs based on one-way rates). At the same time, the carrier is experiencing higher margins through better asset utilization and lower driver turnover through more regular driver schedules.

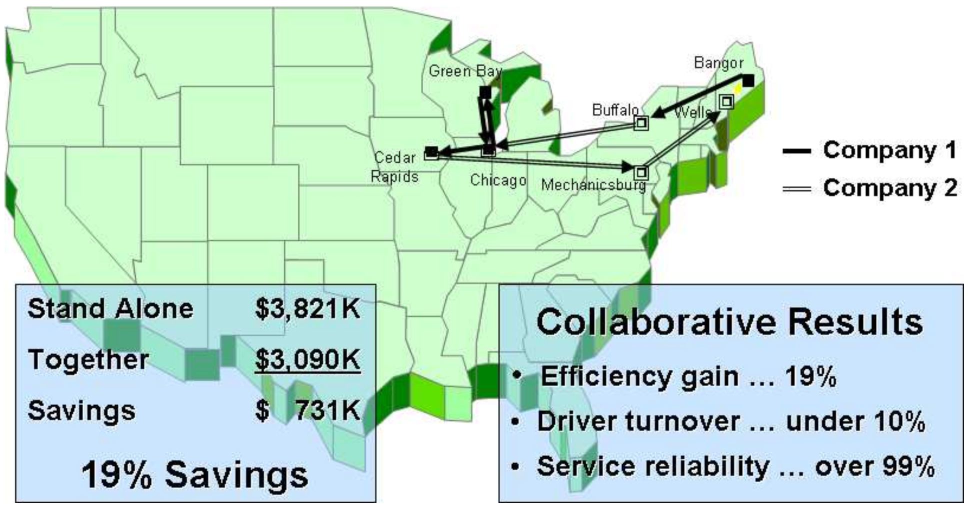


Figure 2.1: Collaborative continuous move example from Nistevo.

Unfortunately, identifying tours minimizing asset repositioning costs in a collaborative logistics network is no simple task. When the number of members of the network, and thus the number of truckload movements to consider, grows, the number of potential tours to examine becomes prohibitively large. In that case, optimization technology is needed to assist the analysts.

In this chapter, we discuss the development of optimization technology that can be used to assist in the identification of repeatable, dedicated truckload tours. This setting is relevant for companies that regularly send truckload shipments, say every day of the week, and are looking for collaborative partners in similar situations.

The underlying optimization problem seeks to find a minimum cost set of tours covering all lanes (regularly scheduled truckload movements) submitted by the member companies of the collaborative logistics network. The problem can be easily formulated as a covering problem on a directed graph, which we will call the Lane Covering Problem (LCP). LCP can be solved efficiently as a minimum cost flow circulation problem.

In practice, there are, more often than not, restrictions limiting the set of acceptable tours, e.g., a restriction on the maximum number of legs that can make up a tour or a restriction on the maximum length or duration of a tour. Therefore, we investigate constrained variants of LCP. Such restrictions make the optimization problem more difficult, both theoretically and practically. We demonstrate that highly effective and extremely efficient optimization-based heuristics can be designed and implemented for these variants as well.

Minimizing asset repositioning is a key ingredient in other truckload transportation procurement studies too. Much of the research reported in the literature, though, has focused on truckload transportation procurement for a single shipper. In that setting, the typical goal is to identify a set of carriers that can serve a shipper's set of lanes at minimal cost. Caplice and Sheffi [15] describe a combinatorial auction run by a shipper to determine the minimum cost allocation of its lanes to carriers. Here it is assumed that the carriers will form bundles of lanes and bid on these bundles given their internal cost structures and the operational synergies the new lanes are creating with their existing network. Song and Regan [70] consider a combinatorial auction model for a shipper procuring transportation services and simulate such an auction to assess the benefits for both the shipper and the carriers. Their study includes a simple model representing the way carriers identify tours minimizing the cost of covering lanes. This model ignores temporal constraints, such as dispatch windows, associated with lanes. Another related stream of research focuses on truckload operations. The objective is to manage the fleet of trucks of a single carrier in response to dynamic demand. For example, Powell et al. [64] develop a stochastic network model for assigning loads to drivers and trucks taking into consideration a high level of demand uncertainty. More recently, an online driver assignment model and the issues in implementing such a model at a carrier are discussed by Powell et al. [83].

2.2. Lane Covering Problems

The core optimization problem, called the lane covering problem (LCP), is stated as follows: given a set of lanes, find a set of tours covering all lanes such that the total cost of the tours is minimized. We define a lane as a regularly scheduled, e.g. daily or weekly, shipment from an origin to a destination. An important assumption we make is that shipment schedules can be adjusted so that the resulting tours can be executed in practice. We also assume that the size of the shipment across each lane is fixed and always equal to a single truckload. In addition, we ignore fleet capacity since the shippers may work with multiple carriers. If all of the lanes in the collaboration, or even in a given tour, cannot be served by a carrier due to fleet size limitations, we assume that the shippers can find additional carriers who are willing to work with the collaborating shippers.

Mathematically, LCP can be stated as follows: given a directed Euclidean graph $D = (V, A)$ with node set V, arc set A, and lane set $L \subseteq A$, find a set of simple directed cycles covering the lanes in L of minimum total length. Note that the cycles do not necessarily have to be disjoint. Let l_{ij} denote the length of arc (i, j) and let x_{ij} be an integer variable indicating how often arc (i, j) is traversed. Assume that arc lengths satisfy the triangle inequality, i.e. $l_{ij} + l_{jk} \leq l_{ik}$ for all arcs $(i, j), (j, k)$ and (i, k). Then the solution to the following minimum cost circulation problem can be decomposed into a set of simple directed cycles covering all lanes with minimum total length:

$$\min \sum_{(i,j)\in A} l_{ij}x_{ij}$$

$$s.t. \quad \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = 0 \quad \forall i \in N$$

$$x_{ij} \geq 1 \quad \forall (i,j) \in L$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in A \setminus L.$$

In our problem definition, we have assumed that a single truckload has to be moved across each lane. It is simple to handle shipments consisting of multiple truckloads. If v_{ij}

denotes the number of shipments that has to be moved across lane $(i, j) \in L$, then all we have to do is replace $x_{ij} \geq 1 \; \forall(i, j) \in L$ with $x_{ij} \geq v_{ij} \; \forall(i, j) \in L$ in the formulation above.

We have been unable to find any literature specifically dealing with LCP. However, there does exist a body of research on a related covering problem. The cycle covering problem (CCP) looks for a least cost cover of a graph with simple cycles, each containing at least three different edges. This constrained version of the Chinese Postman Problem (CPP) was shown to be NP-hard on general graphs by Thomassen [75] and to be equivalent to the CPP on planar graphs by Guan and Fleischner [36] and Kesel'man [22]. Itai et al. [39] provided an upper bound for CCP on 2-connected unweighted graphs and gave a polynomial time algorithm which finds such a cover. Improvements to this bound were proposed by Bermond et al. [8], Alon and Tarsi [2], Fraisse [33], Jackson [40], and Fan [30] , and a simple heuristic was proposed and tested by Labbe et al. [46]. Most recently, Hochbaum and Olinick [37] developed and tested heuristics for a constrained version of the CCP where no cycle in the cover contains more than a prescribed number of edges.

2.3. Cardinality and Length Constrained Lane Covering Problems

In this section, we study two constrained variants of the lane covering problem: the cardinality constrained lane covering problem (CCLCP), in which the number of arcs in a cycle has to be less than or equal to a prespecified number K, and the length constrained lane covering problem (LCLCP), in which the length of a cycle has to be less than or equal to a pre-specified bound B. In both of these variants, we assume that the shipment schedules are adjustable just like we assume in the basic LCP.

Theorem 2.1 *LCLCP is NP-hard.*

Proof: The proof is by reduction from 3-Partitioning.

3-Partitioning. *Given a set* $S = \{a_1, a_2, ..., a_{3n}\}$ *with* $\frac{1}{4}B < a_i < \frac{1}{2}B$ *for* $i = 1, ..., 3n$ *and* $B = \sum_{i=1}^{3n} a_i/n$ *decide whether there exist a partitioning of* S *into distinct subsets* S_k *for* $k = 1, ..., n$ *such that* $\sum_{i \in S_k} a_i = B$.

Construct a directed graph $D = (V, A)$ as follows. For each element i create three arcs e^i_1, e^i_2, e^i_3 with length a_i (the "element" arcs). The arcs are placed in three layers, L_1, L_2, and L_3. Every element arc e^i_k is connected to element arc $e^j_{(k \; mod \; 3)+1}$ for all $j \neq i$ for $k = 1, 2, 3$ by an arc of length $K >> B$ (the "connection" arcs). Observe that:

- Any cycle with length less than $B + 3K$ will have at most six arcs; three element arcs and three connection arcs.
- Any cycle with at most six arcs can only contain a single element arc from among $\{e^i_1, e^i_2, e^i_3\}$.
- Any cycle with at most six arcs contains element arcs corresponding to three different elements a_i, a_j, and a_k.
- Any set $\{a_i, a_j, a_k\}$ can be covered by three "complementing" cycles, i.e., $\{e^i_1, e^j_2, e^k_3\}$, $\{e^j_1, e^k_2, e^i_3\}$, and $\{e^k_1, e^i_2, e^j_3\}$.

It follows easily from these observations that there exists a cycle cover of cost $3n(B+3K)$ with cycles of length less than $B + 3K$ if and only if the instance of 3-partitioning has a feasible solution. $\square$

Theorem 2.2 *CCLCP is NP-hard.*

Proof: This follows immediately from the previous proof by replacing each arc e^k_i of length a_i by a_i copies of length 1 and noticing that 3-Partition is strongly NP-complete. $\square$

2.3.1 Solution Approaches

For the remainder of the section, we concentrate on the solution of CCLCP. However, all the ideas and techniques presented can easily be extended for LCLCP.

We focus on developing highly effective but extremely efficient heuristics as instances encountered in practice are expected to be large.

Let $\mathcal{C}_K = \{C_i, C_2, ..., C_n\}$ represent the set of all directed cycles in D of cardinality less than or equal to K covering at least one lane. Let c_i denote the cost of cycle C_i, let $a_{i\ell}$ be

1 if lane ℓ is covered by cycle C_i and 0 otherwise, and let x_i be a 0-1 variable indicating whether or not cycle C_i is selected to be part of the cycle cover or not. Then CCLCP can be formulated as a set covering problem as follows

$$\min \sum_i c_i x_i$$

$$\sum_i a_{i\ell} x_i \geq 1 \quad \forall \ell \in L$$

$$x_i \in \{0, 1\} \quad \forall i.$$

Note that if a lane is covered by more than one cycle, then in all but one cycle the corresponding arc of the cycle represents empty repositioning (or deadheading). Consequently, the desirability or value of each individual cycle can only be established after the set covering problem has been solved. This is an undesirable feature of the set covering formulation because we plan to develop greedy heuristics for its solution. By explicitly defining the role of an arc in a cycle, i.e., whether it covers a lane or whether it represents deadheading, we can get around this issue at the expense of a larger set of cycles. For each cycle appearing in the set covering formulation, we add all the cycles that can be obtained by replacing one or more arcs covering a lane with arcs representing a deadhead as long as there remains at least one arc covering a lane and there are no consecutive arcs representing deadheads. The process is illustrated in Figure 2.2. If the cycle in Figure 2.2(a) represents one of the original cycles, then the formulation should also include the cycles in Figure 2.2(b). With

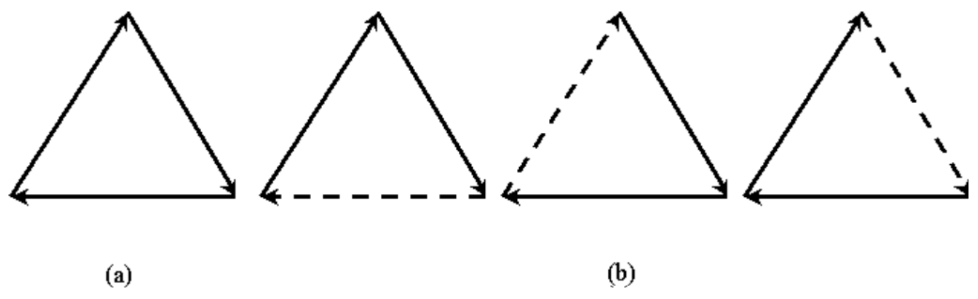


Figure 2.2: Cycle generation example

the new set of cycles CCLCP can be formulated as a set partitioning problem as opposed to a set covering problem. Furthermore, the desirability or value of each individual cycle can be established upfront, because we know precisely which arcs are used to cover lanes and which arcs are used as deadheads. A natural way to represent the desirability of a cycle is to look at the *cover ratio*, defined as the ratio of the length of the lanes covered by the cycle and the total length of the cycle. Note that the cover ratio takes on values in (0,1] and a higher value indicates a more desirable cycle.

Greedy heuristics

Rather than relying on integer programming technology to select a minimum cost set of cycles covering all lanes, which is computationally prohibitive for most practical instances as the number of cycles becomes too large, we implement several greedy selection heuristics. Computational experiments demonstrate that these greedy heuristics are able to produce high quality solutions very efficiently. All greedy selection heuristics are based on the cover ratio, i.e.,

$$\rho_i = \frac{\sum_{a \in C_i \cap L} l_a}{\sum_{a \in C_i} l_a}.$$

The basic greedy heuristic iteratively selects a cycle with the highest cover ratio until all lanes have been covered. Note that this heuristic can be implemented more efficiently using the set of cycles generated for the set partitioning formulation than using the set of cycles generated for the set covering formulation. For the former, we have to sort the set of cycles and, after selecting a cycle, we have to delete all cycles that cover one or more lanes of the selected cycle. For the latter, we have to sort the set of cycles and, after selecting a cycle, we have to update the cover ratio of all cycles that cover one or more lanes in the selected cycle and resort the set of cycles. Given that the set of cycles is likely to be huge, we want to avoid sorting the set of cycles in every iteration.

The basic greedy heuristic selects the cycle with the highest cover ratio regardless of the number of lanes in the cycle. However, we observed that in optimal solutions (to either the set covering or set partitioning problem) cycles covering only a few lanes were dominant. A possible explanation for this phenomenon is that selecting a cycle covering many lanes may cause many other desirable cycles to become less desirable, which, in turn, may lead to some lanes being covered only by undesirable cycles, i.e., cycles with a low cover ratio.

Therefore, we implemented a variant of the basic greedy heuristic in which a modified cover ratio is used to select cycles. Let k_i denote the cardinality of cycle C_i and let d_i denote the number of deadheads in cycle C_i. Then, the modified cover ratio of cycle C_i is defined as

$$\rho_i + \frac{\rho_i - 0.5}{k_i - d_i}.$$

Note that the modified cover ratio favors cycles with high cover ratios and few lanes. We perturb the cover ratio, ρ_i, of each cycle C_i by adding a positive amount which is inversely proportional to the number of lanes, $k_i - d_i$, the cycle covers. This way, a cycle with a few lanes may have a higher *modified ratio* than a cycle with more lanes even though the cycle with more lanes has the higher *ratio*. Subtracting 0.5 from the cover ratio ensures that cycles covering a single lane are selected last. Without subtracting 0.5, these undesirable cycles would have the highest possible value of the perturbation amount.

We have observed that there can be a large disparity between the number of cycles with high cover ratio covering a lane. In other words, for some lanes there are many cycles with a high cover ratio covering the lane, whereas for other lanes there are few, if any, cycles with a high cover ratio covering the lane. Selecting the cycle with highest cover ratio among all possible cycles may result in the deletion of the cycles with highest cover ratio covering some other lanes. These lanes then have to be covered by cycles with low cover ratio. Therefore, we implemented a variant of the basic greedy heuristic using a maximum regret selection criterion. The regret value for a lane ℓ is defined as follows. Let ρ^1_ℓ, be the cover ratio of the cycle covering ℓ with the highest cover ratio and let ρ^2_ℓ be the cover ratio of the cycle covering ℓ with the second highest cover ratio. The regret value is set to $\rho^1_\ell - \rho^2_\ell$. The heuristic first selects the lane that is covered next, i.e., the one with the highest regret value, and then selects the cycle with the highest cover ratio covering that lane.

Speed-ups

A possibility to reduce the computational requirements is to limit the number of cycles generated. Intuitively, cycles with a few arcs representing deadheads are more desirable than cycles with many arcs representing deadheads. Therefore, a simple scheme to generate only a limited number of desirable cycles is to restrict the number of arcs representing a deadhead in a cycle to some number $d \geq 0$. Note that with $d = \lfloor \frac{K}{2} \rfloor$ we obtain all cycles since it is never beneficial to have two consecutive arcs representing deadheads. As we will see in Section 2.3.2, cycles with at most one arc representing a deadhead can be generated extremely efficiently, but, as expected, the quality of a cycle cover consisting only of cycles with at most one arc representing a deadhead is relatively weak. To obtain a lower cost cycle without having to generate all cycles with two or more arcs representing deadheads, we have developed a powerful iterative improvement scheme that merges cycles from a cycle cover consisting only of cycles with at most one arc representing a deadhead (resulting in cycles with two deadheads).

Note that cycle C with one arc representing a deadhead is the union of a path, P, consisting of all lane arcs, with a deadhead arc connecting head of P with its tail. Given two cycles $C_1 = P_1 \cup (head(P_1), tail(P_1))$ and $C_2 = P_2 \cup (head(P_2), tail(P_2))$, each with a single deadhead, we construct a new cycle C_{12} with two deadheads by deleting arcs $(head(P_1), tail(P_1))$ and $(head(P_2), tail(P_2))$ and adding arcs $(head(P_1), tail(P_2))$ and $(head(P_2), tail(P_1))$ (see Figure 2.3).

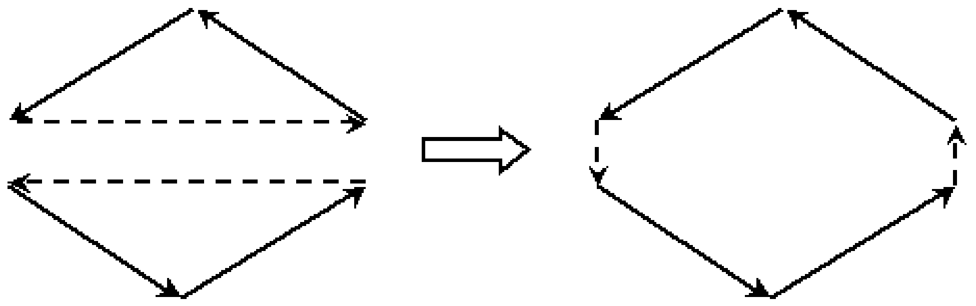


Figure 2.3: Example of merging two single deadhead cycles

Such a merge reduces the cost of the current cycle cover if $c_1 + c_2 > c_{12}$. Given an initial cycle cover $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$, the optimal set of cycle merges can be obtained by finding a minimum cost matching on the digraph $D_{\mathcal{C}} = (V_{\mathcal{C}}, A_{\mathcal{C}})$ where $V_{\mathcal{C}} = \{1, 2, ..., N\}$, $A_{\mathcal{C}} = \{(i,j) : c_i + c_j > c_{ij}\}$ with arc weights $w_{ij} = c_{ij} - (c_i + c_j)$ for $(i,j) \in A_{\mathcal{C}}$. Unfortunately, finding an optimal matching on $D_{\mathcal{C}}$ becomes computationally expensive when $D_{\mathcal{C}}$ gets larger. Hence, we have also developed an effective and efficient *greedy merge heuristic*.

It is straightforward to develop a local search heuristic for solving a minimum cost matching problem. One possibility is to examine, in some order, all possible cycle merges until an improving cycle merge is found, implement that cycle merge and set the new cycle aside, and continue. This heuristic examines each possible combination of cycles at most once, and is thus very efficient, but may not lead to a significant cost reduction. Our greedy merge heuristic first identifies all beneficial cycle merges, finds the most improving cycle merge, implements that cycle merge, sets the resulting cycle aside, and then repeats.

The efficiency of the greedy merge heuristic is further improved by exploiting the following claim.

Claim 2.3 *If merging cycles C_1 and C_2 leads to a cost reduction, then at least one of the following two inequalities holds:*

1. $\ell_{(head(P_1),tail(P_1))} > \ell_{(head(P_1),tail(P_2))}$
2. $\ell_{(head(P_2),tail(P_2))} > \ell_{(head(P_2),tail(P_1))}$.

Proof: Assume that neither of the inequalities holds. If $c_1 + c_2 > c_{12}$, then

$$\ell_{(head(P_1),tail(P_1))} + \ell_{(head(P_2),tail(P_2))} > \ell_{(head(P_1),tail(P_2))} + \ell_{(head(P_2),tail(P_1))},$$

which gives a contradiction. □

Hence the number of cycle merges that needed to be examined for identifying the set of beneficial ones may be significantly reduced.

2.3.2 Implementation

2.3.2.1 Cycle Generation

Given a directed Euclidean graph $D = (V, A)$ with node set V, arc set A, and lane set $L \subseteq A$, we have to generate all possible simple cycles with K or fewer arcs covering at least one lane. Two consecutive arcs representing deadheads are not allowed because they can be replaced by a single arc representing a shorter (or at least no longer) deadhead.

To be able to control the number of cycles that are generated, we impose an additional limit d on the number of arcs representing deadheads in a cycle. Limiting the number of arcs representing deadheads provides a simple mechanism to generate only cycles that are likely to be of high quality.

Generating cycles with at most one arc representing a deadhead

Observe that any cycle of cardinality k with a single arc representing a deadhead contains a simple path of cardinality $k-1$ consisting only of arcs covering lanes. The single arc representing the deadhead connects the end of the path covering lanes with the beginning of that path. Consequently, generating cycles with one arc representing a deadhead is equivalent to generating paths in the subgraph defined by the lanes set (the so-called lane graph) and connecting the end of the path to the start of the path by a deadhead. Generating paths in the lane graph is simple if the lane graph is stored using a forward and reverse star representation. The other issue that needs to be resolved is how to avoid generating duplicate cycles. Generating multiple copies of a cycle is a waste of memory and computational effort since at most one of the copies can appear in the final solution. We have experimented with two schemes that avoid the generation of duplicate cycles:

1. Generate all cycles containing the arc that covers a specific lane, delete the lane from the lane graph and repeat until the lane graph is empty.

2. Generate all cycles containing the arc that covers a specific lane as the first arc of the path consisting of arcs covering lanes. Repeat for all lanes in the lane graph.

Both schemes can easily and efficiently be implemented using recursive procedures SearchForward and SearchBackward working on the lane graph. The first scheme is described in more detail below (the second scheme follows easily).

SearchForward receives as input a *base path* of length less than or equal to $K-1$. SearchForward starts by creating the cycle consisting of the base path and a single deadhead connecting the end of the base path with its beginning. Next, if the base path has length strictly less than $K-1$, then we consider expanding the base path by adding arcs at its end, i.e., arcs with tail node equal to the head of the base path (which can be done efficiently using the forward star representation of the lane graph). If the head of the arc being consider for addition to the base path is some other node on the base path, then that arc is ignored. For all other arcs, the base path is expanded and SearchForward is called recursively.

Algorithm 2.1 SearchForward(P)

```
Create cycle with deadhead: $P \cup (head(P), tail(P))$
if $length(P) < K - 1$ then
  for all arcs $a$ in forward star($head(P)$) do
    if $head(a) \notin nodes(P)$ then
      SearchForward($P \cup a$)
    end if
  end for
end if
```

SearchBackward receives as input a base path of length less than or equal to $K - 1$. We consider expanding the base path by adding arcs at its beginning, i.e., arcs with head node equal to the tail of the base path (which can be done efficiently using the reverse star representation of the lane graph). If the tail of the arc being considered for addition to the base path is equal to the head of the base path, then we have identified a cycle without deadheads. If the tail of the arc being considered for addition to the base path is some other node on the base path, then that arc is ignored. For all other arcs, if the length of the base path is strictly less than $K - 1$, the base path is expanded and SearchForward is called followed by a recursive call to SearchBackward.

Algorithm 2.2 SearchBackward(P)

```
for all arcs $a$ in reverse star($tail(P)$) do
  if $tail(a) = head(P)$ then
    Create cycle without deadhead: $(tail(a), head(P)) \cup P$
  end if
  if $length(P) < K - 1$ and $tail(a) \notin nodes(P)$ then
    SearchForward($a \cup P$)
    SearchBackward($a \cup P$)
  end if
end for
```

Given procedures SearchForward and SearchBackward all cycles of cardinality less than or equal to K with at most one arc representing a deadhead can be generated using Algorithm 2.3.

Generating cycles with more than one arc representing a deadhead

Algorithm 2.3 Cycle generation

```
for $\ell \in L$ do
  SearchForward($\ell$)
  SearchBackward($\ell$)
  Delete $\ell$ from the lane graph
end for
```

Generating cycles with more than one arc representing a deadhead proceeds along the same lines. However, since the maximum number of arcs representing deadheads d in a cycle is greater than one, the base path may contain up to $d-1$ arcs representing deadheads. In the case where at most one arc representing a deadhead was allowed, both the forward and the backward search procedures expanded the base path using only arcs representing lanes that were incident to an endpoint of the path. When more arcs representing deadheads are allowed, the base path is also expanded with arcs representing lanes whose endpoints are not incident to an endpoint of the base path. When an arc representing a deadhead is introduced in the base path, we go through all the nodes in the graph that are not part of the base path and expand the base path using arcs from the lane graph incident to those nodes. Note that by doing so, we increase the length of the base path by two arcs and that the expanded base path has one more arc representing a deadhead than the base path. It is only necessary to allow incorporation of arcs representing deadheads in the base path in one of the procedures SearchForward and SearchBackward. We chose to allow deadhead-increasing path expansions in the forward search only.

It should be clear that because we cannot work exclusively on the lane graph when constructing paths, generating cycles with more than one arc representing deadheads is not as efficient as generating cycles with at most one arc representing deadheads.

2.3.2.2 Greedy Heuristics

The crucial and most time consuming step of the basic greedy heuristic is "sorting" the cycles in order of their cover ratio. In fact, it is not necessary to sort the cycles, because we only need to be able to identify the cycle with the highest cover ratio. Therefore, we maintain a binary heap of the cycles with the cover ratio as the key. The greedy heuristic repeatedly takes the top element off the heap until the top element corresponds to a cycle which has not yet been deleted (has not yet been marked as deleted, to be more precise). The selected cycle is added to the solution and all other cycles covering any of its lanes are deleted. This process is repeated until all lanes are covered.

The implementation of the maximum regret variant of the greedy heuristic requires the use of more than one heap. For each lane, we maintain a heap of the cycles covering that lane with the cover ratio as the key. These heaps are used to find the best (highest cover ratio) and the second best (second highest cover ratio) cycles covering the lanes. The lanes themselves are also kept in heap with their regret value, i.e., the difference between the cover ratio of the best and the second best cycle, as the key. The greedy heuristic repeatedly takes the top element of the lane heap. The best cycle for that lane is selected to be part of the solution. Each cycle selection is again followed by the deletion of cycles covering any of its lanes. The set of deleted cycles may include the best or the second best cycle of some of the as of yet uncovered lanes. In that case, the regret values for these lanes change and the lane heap needs to be updated. In order to be able to update the heap, we maintain a set of pointers linking lanes to the heap elements corresponding to these lanes.

2.3.2.3 Greedy Merge Heuristic

The greedy merge heuristic starts with a cycle cover consisting of cycles with at most one arc representing a deadhead and goes through these cycles in some arbitrary order, identifying all the beneficial merge opportunities for the cycle in hand. We will refer to the cycle in hand as the *base cycle* and will denote it by C_b. The algorithm only evaluates merging the base cycle with cycles C_j for which $tail(P_j)$ is within distance $\ell_{(head(P_b),tail(P_b))}$ of $head(P_b)$. If merging C_b with C_j is beneficial then (b, j) is stored in a heap data structure, $ImprovementHeap$, with $c_{bj} - (c_b + c_j)$ as key. Then iteratively the best merge opportunity is popped up from $ImprovementHeap$. If the cycles involved are not marked as already merged, then they are merged and marked as merged.(See Algorithm 2.4 for a detailed description.)

We have shown that restricting the distance between the first node of the path of a candidate cycle and the last node of the path of the base cycle is sufficient to find all the

Algorithm 2.4 GreedyMergeHeuristic($\hat{C}$)

```
for all cycles $C_i$ in $\hat{C}$ do
  for all cycles $C_j$ in $\hat{C}$ with $\ell_{(head(P_i),tail(P_i))} > \ell_{(head(P_i),tail(P_j))}$ do
    if $c_{ij} < c_i + c_j$ then
      $Insert((i, j), ImprovementHeap)$
    end if
  end for
end for
while $ImprovementHeap \neq \emptyset$ do
  $(k, l) := Pop(ImprovementHeap)$
  if cycles $C_k$ and $C_l$ are not marked as merged then
    merge $C_k$ with $C_l$ and mark them as merged
  end if
end while
```

improving merges. A geometric data structure, the *k-d tree*, allows for efficiently performing such queries. See Bentley [7] for more information on *k-d tree* trees.

2.3.3 Computational Experiments

The goal of the computational study reported on in this section was to demonstrate the effectiveness and efficiency of the algorithms presented above.

The algorithms have been tested on randomly generated Euclidean instances. Instance generation is controlled by the following parameters: the number of points, the number of lanes, and the number of clusters. Clusters are introduced to represent geographical concentrations of points, such as metropolitan areas. First, the appropriate number of points is randomly generated within a $2,000 \times 2,000$ square and the complete digraph defined by these points is formed. Next, the desired number of lanes is randomly selected from among the arcs of the complete digraph ensuring that each point has at least one lane incident to it.

To properly analyze the performance of the algorithms, we have generated instances for several different parameter settings. We have varied the number of points, the number of lanes, and the number of clusters. More specifically, we generated instances with 100, 200, 300, 400 and 500 points, with an average number of lanes incident to a point of two or five, and with no clusters or five clusters. A summary of the different parameters settings used to create the instances is given in Table 2.1.

Table 2.1: Parameter settings for the instances.

Instance	#Points	#Lanes	#Clusters
1	100	200	0
2	100	200	5
3	200	400	0
4	200	400	5
5	300	600	0
6	300	600	5
7	400	800	0
8	400	800	5
9	500	1000	0
10	500	1000	5
11	100	500	0
12	100	500	5
13	200	1000	0
14	200	1000	5
15	300	1500	0
16	300	1500	5
17	400	2000	0
18	400	2000	5
19	500	2500	0
20	500	2500	5

For each parameter setting we generated five instances. As the computational behavior varied little for instances generated with the same parameter settings, we present the computational results for actual instances (for the first out of the five) rather than averages over all five instances. In all our experiments we restricted the cardinality of a cycle to at most five. Therefore, by generating all cycles with at most two arcs representing repositioning we generate all cycles, since the underlying digraph is Euclidean and complete. All computational experiments were performed on a computer with a 2.4 GHz Intel Xeon processor, 2 GB of memory, and running Linux operating system kernel version 2.4.18-24.8.0. All integer programs were solved using Xpress-Optimizer version 14.27.

The first set of experiments focuses on the quality of the solutions produced by the greedy heuristics. We compare the values of the solutions produced by the greedy heuristics for a given set of cycles to the value of the optimal solution produced by solving the set partitioning formulation over the same set of cycles. The results can be found in Table 2.2, where we present for each instance the number of cycles generated (**#Cycles**), the optimal value (**IP cost**), and for each of the greedy heuristics the percentage increase in solution value over the optimal value (**Greedy**, **Modified**, and **Regret**, respectively). Table 2.2(a) contains the results for the experiments with cycles with at most one deadhead arc, and Table 2.2(b) contains the results for the experiments with all cycles.

Several observations can be made when analyzing the results of this experiment. First, as expected, the number of cycles grows rapidly with the number of lanes, especially when the number of arcs allowed to represent repositioning increases (with at most one deadhead arc allowed the number of cycles remains manageable, but with more than one deadhead arc allowed the number of cycles becomes prohibitively large quickly). Second, we observe that the different variants of greedy perform similarly, with a slight edge for the regret variant when we consider cycles with at most one deadhead arc and with a slight edge for basic greedy when we consider all cycles. More interesting may be the fact that when we consider cycles with at most one deadhead arc the percentage increase over the optimal value is more than 10%, whereas when we consider all cycles the percentage increase over the optimal value is less than 3%, with the basic greedy heuristic even less than 2%. Finally, we observe that the optimal value increases by 5% on average when cycles with at most one deadhead are considered (as opposed to considering all cycles).

Table 2.3 shows the computation times (in CPU seconds) for the first set of experiments, i.e., the experiments for which results were reported in Table 2.2.

We observe that optimization, in the form of solving the set partition formulation, is computationally prohibitive, even when we limit ourselves to cycles with at most one deadhead arc. For the instances with 500 points and 2500 lanes, the solution of the integer program takes about two days of CPU time. On the other hand, the greedy heuristics are very efficient. When considering cycles with at most one deadhead all variants solved all instances in less than one second. When considering all cycles computation times start to increase (as the number of cycles increases dramatically), but are still well within acceptable limits with about 60 seconds for instances with 400 points and 2000 lanes. Among the heuristics, the regret variant requires the least amount of computation time. The difference in the computation time is due to the fact that basic greedy and modified greedy maintain a heap of all cycles, which has to be updated each time the top element is removed, whereas the regret variant maintains several heaps of smaller size and only a subset of these heaps

Table 2.2: Quality of solutions produced by the greedy heuristics.

(a) Cycles with at most one deadhead.

Instance	#Cycles	Greedy	Modified	Regret	IP Cost
1	2,575	9.39	10.24	8.34	414,070
2	2,834	7.56	12.68	9.64	72,263
3	5,088	9.27	9.38	8.39	858,564
4	5,364	14.72	15.02	10.37	145,516
5	8,849	9.44	10.57	8.27	1,241,191
6	9,299	11.63	14.24	9.02	206,970
7	12,365	9.82	9.98	10.69	1,665,048
8	10,401	10.13	10.22	10.20	301,751
9	14,957	9.89	11.14	9.49	2,108,771
10	13,998	15.44	13.84	9.74	367,076
11	63,306	10.92	11.40	9.01	917,229
12	73,293	11.26	6.38	8.84	162,233
13	133,528	13.00	12.85	10.92	1,920,295
14	139,088	15.69	10.84	12.25	341,879
15	204,547	12.93	12.94	9.75	2,902,527
16	213,813	14.86	11.72	10.40	509,450
17	288,084	13.57	12.75	11.47	3,842,429
18	275,473	14.55	11.36	11.85	691,733
19	348,581	13.25	12.60	10.49	4,811,773
20	342,551	15.52	11.58	12.84	844,007
Average		12.14	11.59	10.10	

(b) All cycles.

Instance	#Cycles	Greedy	Modified	Regret	IP Cost
1	68,411	3.24	4.13	5.81	389,623
2	71,392	0.52	2.01	1.93	71,368
3	299,544	3.34	4.22	3.68	808,845
4	292,370	0.56	1.64	1.44	139,728
5	708,160	3.10	4.59	4.33	1,153,373
6	704,910	0.43	1.41	1.22	197,536
7	1,265,106	2.88	4.52	4.67	1,542,209
8	1,158,255	0.35	1.23	2.28	282,988
9	1,993,848	2.74	4.44	3.40	1,948,477
10	1,958,168	0.46	1.12	1.93	347,562
11	1,024,911	2.20	3.44	3.35	916,591
12	1,062,377	0.30	1.86	1.82	162,187
13	4,382,831	2.30	4.21	2.93	1,903,768
14	4,320,643	0.25	1.19	0.92	341,660
15	9,971,238	IP solver out of memory			
16	9,847,572	IP solver out of memory			
17	17,887,918	IP solver out of memory			
18	17,272,007	IP solver out of memory			
19	$27,959,929^a$	Cycle generation out of memory			
20	$27,244,261^a$	Cycle generation out of memory			
Average		1.62	2.86	2.84	

^{a}: Number of cycles generated at the time memory ran out.

Table 2.3: Computation times of the greedy heuristics.

(a) Cycles with at most one deadhead.

Instance	Cycle Generation	Greedy	Modified	Regret	IP
1	0.00	0.00	0.01	0.00	0
2	0.00	0.00	0.00	0.01	0
3	0.00	0.00	0.00	0.01	0
4	0.00	0.01	0.01	0.01	0
5	0.00	0.01	0.01	0.01	6
6	0.00	0.01	0.02	0.01	8
7	0.00	0.02	0.01	0.02	19
8	0.01	0.01	0.01	0.01	0
9	0.00	0.02	0.02	0.02	11
10	0.01	0.02	0.02	0.01	1
11	0.02	0.11	0.10	0.07	898
12	0.03	0.12	0.12	0.10	1,453
13	0.05	0.26	0.26	0.17	7,286
14	0.06	0.26	0.26	0.18	8,764
15	0.08	0.44	0.43	0.29	26,656
16	0.08	0.45	0.45	0.30	30,844
17	0.12	0.66	0.65	0.45	79,707
18	0.11	0.61	0.61	0.41	73,200
19	0.14	0.83	0.82	0.56	171,587
20	0.15	0.79	0.80	0.55	164,039

(b) All cycles.

Instance	Cycle Generation	Greedy	Modified	Regret	IP
1	0.04	0.10	0.10	0.06	115
2	0.04	0.11	0.11	0.05	83
3	0.19	0.63	0.62	0.26	2,336
4	0.19	0.58	0.59	0.24	1,494
5	0.45	1.69	1.72	0.70	10,445
6	0.45	1.66	1.68	0.68	9,692
7	0.82	3.31	3.33	1.32	39,538
8	0.76	2.93	2.96	1.21	26,906
9	1.30	5.63	5.66	2.20	82,547
10	1.27	5.45	5.46	2.13	125,904
11	0.65	2.59	2.63	1.05	26,715
12	0.67	2.67	2.70	1.05	20,382
13	2.73	13.78	13.85	4.91	87,739
14	2.76	13.34	13.51	4.95	121,149
15	6.31	35.54	35.92	13.10	–
16	6.31	34.31	34.75	12.70	–
17	11.49	68.02	68.67	24.68	–
18	11.25	63.51	64.71	24.21	–
19	17.97	–	–	–	–
20	17.63	–	–	–	–

has to be updated at each iteration.

The previous experiments have clearly shown that the considered set of cycles impacts both the quality of the solution as well as the efficiency with which a solution can be obtained. The mechanism used to control the set of cycles is the maximum number of arcs allowed to represent repositioning. We have seen that we can generate cycles with at most one deadhead arc very efficiently. We have also seen that allowing more than one deadhead arc in a cycle results in higher quality solutions, but at the expense of sometimes prohibitively large time and memory requirements as the numbers of cycles being generated explodes. In order to take advantage of the efficiency of generating cycles with at most one deadhead, we first create a solution consisting of cycles with at most one deadhead arc and then improving the resulting cycle cover by merging cycles.

The second set of experiments investigates the merits of the Greedy Merge heuristic, which can be started from any feasible cycle cover. Initial experimentation revealed that the quality of the cycle cover produced by the Greedy Merge heuristic when started from the cycle covers produced by the three greedy selection heuristics was about the same, with slightly better results when the basic greedy selection heuristic was used. (Note that the quality of the cycle covers produced by the basic greedy selection heuristic when considering only cycles with at most one deadhead was slightly worse than the quality of the cycle covers produced by the two other variants.) Table 2.4 presents the percentage improvement obtained by the Greedy Merge heuristic as well as the maximum possible percentage improvement computed by solving a weighted matching problem on the complete graph with vertices corresponding to the cycles in the initial cycle cover and edge weights corresponding to the value of merging the two cycles represented by the incident vertices. More precisely, we report for each instance the number of cycles with at most one deadhead (**#Cycles Generated**), the number of cycles selected by the greedy heuristic to form the initial cycle cover (**#Cycles Selected**), the number of evaluated cycle merges (**#Merges Evaluated)**, and the number of improving cycle merges found among the evaluated cycle merges (**#Improving Merges**), followed by the time required (**Time**), the percentage improvement (**%Improve**), and the number of cycles in the resulting cycle cover (**#Cycles**) for both the Greedy Merge heuristic and the weighted matching based algorithm. Note that the number of evaluated cycle merges is less than the total number of cycle pairs as we use Claim 2.3 to reduce the number of cycle pairs to evaluate.

Several observations can be made when analyzing the results of this experiment. First, the Greedy Merge heuristic is very effective as the improvements obtained are close to the maximum possible improvements. Furthermore, the actual improvements are significant; on average more than 11%. Second, the Greedy Merge heuristic is very efficient as it required less than 1 second for all instances. (For the considered instances solving a matching problem is still computationally feasible, although we see that the computational effort starts to increase for the larger instances.) Finally, we observe that the use of Claim 2.3 substantially reduces the number of cycle pairs evaluated (by more than half), and thus, in conjunction with the use of k-d trees, the solution time.

When we refer to the Greedy Merge heuristic in our discussion of the results presented in Table 2.5, we refer to the algorithm that generates all cycles with at most one deadhead arc, uses the basic greedy heuristic to construct an initial cycle cover, and then uses the Greedy Merge heuristic to improve the initial cycle cover by greedily merging cycles. Table 2.5 presents maybe even more important results concerning the Greedy Merge heuristic as it compares the values of the cycle covers obtained with the Greedy Merge heuristic to the values of the cycle covers obtained with the basic greedy heuristic when all cycles are considered. Table 2.5 presents for each instance the solution time (**Time**), the cost of the cycle cover produced (**Cost**), and the cover ratio (**Ratio**), for both the basic greedy heuristic and the Greedy Merge heuristic. The cover ratio of a solution (**Ratio**) is computed by dividing the sum of the lengths of the lanes by the cost of the cycle cover produced (**Cost**). The results clearly demonstrate that the Greedy Merge heuristic is able to construct high quality solutions in a short amount of time; all cycle covers were computed in less than 2 seconds. Also, for the instances for which we were able to obtain optimal solutions, the values of the solutions produced by the Greedy Merge heuristic were never more than 2.5% higher.

These results suggest that the Greedy Merge heuristic can be used to solve much larger

Table 2.4: Improvements by merging cycles.

	#Cycles	#Cycles	#Merges	#Improving	Greedy Merge			Matching Merge		
Instance	Generated	Selected	Evaluated	Merges	Time	%Improve	#Cycles	Time	%Improve	#Cycles
1	2,575	95	2,750	699	0	10.24	69	0	10.63	68
2	2,834	83	1,690	272	0	7.30	65	0	7.31	65
3	5,088	185	10,184	2,386	0	10.39	132	0	10.79	132
4	5,364	194	11,182	2,732	0	15.01	137	0	15.11	136
5	8,849	284	22,877	5,990	0	11.51	200	0	12.08	199
6	9,299	266	19,042	4,214	0	13.33	197	0	13.42	195
7	12,365	368	39,343	9,469	0	12.39	265	0.2	12.94	259
8	10,401	370	42,592	10,117	0	14.04	263	0.2	14.13	264
9	14,957	452	62,447	14,630	0	11.99	328	0.3	12.40	324
10	13,998	472	65,749	16,226	0	16.60	336	0.4	16.72	334
11	63,306	202	7,067	1,833	0	7.91	155	0	8.12	156
12	73,293	197	6,489	1,611	0	9.40	152	0	9.47	154
13	133,528	403	29,716	8,972	0	10.08	301	0.2	10.42	296
14	139,088	412	36,017	10,696	0	13.23	304	0.2	13.27	302
15	204,547	597	71,727	20,453	0	9.88	445	0.6	10.35	443
16	213,813	606	73,859	20,714	0	12.30	450	0.6	12.49	448
17	288,084	800	128,555	34,811	0.1	10.99	600	1.5	11.36	593
18	275,473	812	142,850	38,156	0.1	12.05	605	1.9	12.10	601
19	348,581	978	205,385	53,539	0.1	10.75	735	2.9	11.15	733
20	342,551	1,019	202,834	57,064	0.1	13.18	749	4	13.22	749

Table 2.5: Greedy heuristic with all cycles vs. Greedy Merge heuristic.

	Greedy with all cycles			Greedy Merge		
Instance	Time	Cost	Ratio	Time	Cost	Ratio
1	0.14	402,265.75	0.84	0.00	406,594.24	0.83
2	0.15	71,736.42	0.88	0.00	72,047.59	0.88
3	0.82	835,887.82	0.86	0.00	840,684.59	0.86
4	0.77	140,510.54	0.92	0.01	141,881.13	0.91
5	2.14	1,189,171.30	0.88	0.01	1,201,909.66	0.87
6	2.11	198,389.42	0.93	0.01	200,233.67	0.92
7	4.13	1,586,598.80	0.88	0.02	1,601,973.50	0.87
8	3.69	283,991.39	0.92	0.02	285,668.71	0.91
9	6.93	2,001,826.06	0.90	0.02	2,039,424.24	0.88
10	6.72	349,176.35	0.97	0.03	353,386.76	0.96
11	3.24	936,785.41	0.90	0.13	936,978.73	0.90
12	3.34	162,675.18	0.95	0.15	163,522.71	0.95
13	16.51	1,947,611.69	0.91	0.31	1,951,098.51	0.91
14	16.10	342,510.00	0.95	0.32	343,202.71	0.95
15	41.85	2,941,736.22	0.91	0.52	2,954,008.48	0.91
16	40.62	510,102.46	0.97	0.53	513,159.91	0.96
17	79.51	3,859,155.52	0.93	0.88	3,884,093.05	0.92
18	74.76	692,772.21	0.95	0.82	696,940.84	0.94
19	–	–	–	1.07	4,863,680.25	0.93
20	–	–	–	1.04	846,437.07	0.98

instances. To get a better sense of the sizes of instance that can be handled comfortably, we used the Greedy Merge heuristic to solve instances with 7,500 points and 15,000 and 37,500 lanes. The results can be found in Table 2.6. The top portion of Table 2.6 shows the instance characteristics (**#Points**, **#Lanes**, **#Clusters**), the number of cycles with at most one deadhead generated (**#Cycles Generated**), the number of cycles selected to form the initial cycle cover (**#Cycles Selected**), the cover ratio (**Ratio**), and the time required to produce the initial cycle cover, including cycle generation time (**Time**). The bottom portion of Table 2.6 shows the number of merges evaluated (**#Merges Evaluated**), the number of improving merges identified (**#Improving Merges**), followed by the solution time (**Time**), the cover ratio (**Ratio**), and the number of cycles in the final cycle cover (**#Cycles**) for both the Greedy Merge heuristic and the weighted matching based algorithm.

The results in Table 2.6 demonstrate once more that the Greedy Merge heuristic produces high quality solutions with little computational effort even for large instances. Furthermore, it is clear that for instances of this size the weighted matching based approach is no longer viable as computation times reach more than 6 hours.

The final set of computational experiments focuses on a completely different approach

Table 2.6: Greedy Merge heuristic on large instances.

#Points	#Lanes	#Clusters	#Cycles Generated	#Cycles Selected	Ratio	Time
7,500	15,000	0	218,041	7,044	0.76	0.64
7,500	15,000	5	216,018	7,074	0.79	0.62
7,500	37,500	0	5,376,238	15,054	0.82	26.15
7,500	37,500	5	5,332,869	15,073	0.86	25.64

#Merges	#Improving	Greedy Merge			Matching Merge		
Evaluated	Merges	Time*	Ratio	#Cycles	Time*	Ratio	#Cycles
15,864,235	3,835,394	38.7	0.91	4,912	4,026	0.92	4,858
14,554,221	3,512,656	35.2	0.97	4,922	3,713	0.97	4,915
54,290,635	11,222,305	152.6	0.95	11,328	24,561	0.95	11,250
48,361,955	10,313,981	137.4	0.99	11,301	22,360	0.99	11,299

*: Time to find the initial solution not included

to limiting the number of cycles to consider and manipulate. Instead of blindly generating all cycles and using basic greedy to select cycles to form a cycle cover, we generate all cycles, but only keep "good" cycles and use basic greedy to select cycles to form a cycle cover from among these good cycles. The cover ratio is used to determine whether a cycle is a "good" cycle or not. All cycles with a cover ratio below a certain threshold are discarded. To ensure the existence of a feasible cycle cover, we always keep the trivial out-and-back cycles covering a single lane. (The number of such cycles is small as it is equal to the number of lanes.) In the computational experiments we used threshold values of 0.6, 0.7, and 0.8. (Note that a threshold of 0.5 would result in the generation of all the cycles.) The results are divided over Tables 2.7, 2.8, and 2.9. Table 2.7 lists the number of cycles generated, Table 2.8 shows the percentage increase in cost over the solution produced by the basic greedy heuristic when considering all cycles, and Table 2.9 reports the computation times.

Several observations can be made when analyzing the results of this experiment. First, we see that the use of thresholds results in a substantial reduction in the number of cycles generated (even for threshold value 0.6), although the number of generated cycles is still significantly larger than the number of cycles with at most one deadhead. Second, we see that the quality of the solutions produced by the Greedy Merge heuristic is comparable or better than the quality of the solutions produced when using thresholds to limit the number of cycles (even for a threshold value of 0.6). Finally, we note that even though filtering out

Table 2.7: Number of cycles generated for different thresholds.

Instance	#Cycles (Single Deadhead)	#Cycles (All)	Threshold 0.6	0.7	0.8
1	2,575	68,411	34,869	12,160	2,969
2	2,834	71,392	31,376	19,429	10,400
3	5,088	299,544	163,371	54,426	10,226
4	5,364	292,370	131,272	79,811	42,858
5	8,849	708,160	381,357	127,021	23,024
6	9,299	704,910	325,763	187,505	93,663
7	12,365	1,265,106	668,545	215,131	36,371
8	10,401	1,158,255	517,368	318,853	174,089
9	14,957	1,993,848	1,077,048	350,255	58,774
10	13,998	1,958,168	908,360	556,159	290,268
11	63,306	1,024,911	563,839	210,961	55,886
12	73,293	1,062,377	506,657	319,196	172,956
13	133,528	4,382,831	2,467,449	855,757	180,055
14	139,088	4,320,643	2,055,495	1,278,280	686,713
15	204,547	9,971,238	5,604,130	1,953,037	381,166
16	213,813	9,847,572	4,698,448	2,913,706	1,565,409
17	288,084	17,887,918	10,105,878	3,445,337	643,667
18	275,473	17,272,007	8,156,622	4,986,126	2,661,502
19	348,581	$27,959,929^b$	15,781,163	5,316,436	959,017
20	342,551	$27,244,261^b$	13,077,520	8,137,303	4,257,479

^{b}: Number of cycles generated at the time memory ran out

cycles with low cover ratio reduces the memory requirements, all the cycles must still be generated, and, as a result, the Greedy Merge heuristic is much more efficient, requiring only a small fraction of the computation time required by the filtering based approaches.

2.4. Time Constrained Lane Covering Problem

In this section, timing considerations are a key focus of our efforts which are critical to practical viability. We show that a highly effective and extremely efficient heuristic can be designed and implemented. The heuristic incorporates, among others, fast routines for checking time feasibility of a tour in the presence of dispatch time windows and for minimizing the waiting along a tour by appropriately selecting a starting location and departure time. We demonstrate the effectiveness of the algorithms developed on various randomly generated instances simulating real-life supply chain structures, and on instances derived from data obtained from a strategic sourcing consortium for a \$14 billion dollar sized US industry.

As mentioned in the introduction of this chapter, we focus on developing technology

Table 2.8: Percentage increase for different thresholds.

	Threshold			Greedy
Instance	0.6	0.7	0.8	Merge
1	0.80	3.03	9.04	1.08
2	1.72	3.15	2.99	0.43
3	0.76	3.17	5.43	0.57
4	0.45	0.73	0.74	0.98
5	0.61	1.53	6.24	1.07
6	0.36	0.99	1.03	0.93
7	0.43	2.58	6.33	0.97
8	0.20	0.30	0.59	0.59
9	0.21	1.66	6.63	1.88
10	0.23	0.33	0.68	1.21
11	0.68	2.18	5.50	0.02
12	0.10	0.75	1.09	0.52
13	0.48	1.84	5.34	0.18
14	0.37	0.89	1.13	0.20
15	0.49	1.49	5.50	0.42
16	0.18	0.53	0.90	0.60
17	0.13	1.14	4.25	0.65
18	0.47	0.52	0.92	0.60
Average	0.48	1.49	3.57	0.72

Table 2.9: Solution times for different thresholds.

	#Cycles	Threshold			#Cycles
Instance	(All)	0.6	0.7	0.8	(Single Deadhead)
1	0.14	0.08	0.04	0.03	0.00
2	0.15	0.07	0.05	0.03	0.00
3	0.82	0.46	0.20	0.13	0.00
4	0.77	0.37	0.25	0.19	0.01
5	2.14	1.22	0.55	0.33	0.01
6	2.11	1.06	0.70	0.47	0.01
7	4.13	2.33	1.04	0.61	0.02
8	3.69	1.85	1.26	0.88	0.02
9	6.93	3.93	1.75	0.99	0.02
10	6.72	3.42	2.29	1.54	0.03
11	3.24	1.89	0.91	0.53	0.13
12	3.34	1.74	1.19	0.81	0.15
13	16.51	9.71	4.38	2.45	0.31
14	16.1	8.21	5.64	3.88	0.32
15	41.85	24.49	10.88	5.85	0.52
16	40.62	21.00	14.70	9.51	0.53
17	79.51	46.42	20.57	10.88	0.88
18	74.76	38.90	26.23	17.40	0.82
19	17.97^c	75.13	33.52	17.40	1.07
20	17.63^c	64.54	44.17	28.86	1.04

^{c}: Time until memory runs out

for identifying repeatable, dedicated truckload continuous move tours for companies that regularly send truckload shipments and are looking for collaborative partners. Properly incorporating timing considerations in this technology is of critical importance to the practical viability. There are two sources of temporal constraints: dispatch time windows on the loads to be transported and U.S. Department of Transportation (DOT) Hours of Service regulations. Regularly scheduled truckload movements are usually referred to as lanes and are specified by an origin, a destination, and a dispatch window. The dispatch window indicates the time interval in which the load to be moved should be dispatched, for example Mondays between 8am and 2pm. A dispatch window incorporates information on the time at which the load to be moved is ready as well as information on the time at which the load needs to depart to ensure it will reach its destination on time. U.S. DOT Hours of Service regulations limit driving and duty hours of truck drivers. Truck drivers may not drive more than 11 hours following 10 hours off-duty, may not drive beyond the 14th hour after coming on-duty, and may not drive after 60/70 hours on-duty in 7/8 consecutive days. From an algorithmic perspective, dispatch windows pose more interesting challenges.

Consequently, we focus on the Time-Constrained Lane Covering Problem (TCLCP), which we define as follows. For a given set of lanes, find a set of tours covering all lanes such that the total duration of the tours is minimized and the dispatch windows are respected. It is important to note that we do not consider U.S. DOT regulations in TCLCP. A more formal description of TCLCP is as follows: given a complete bi-directed Euclidean graph $D = (V, A)$ with node set V, arc set A, and travel time t_a for each $a \in A$, a period length T, and a lane set L, where each lane $l \in L$ is specified by an arc $a \in A$ and a dispatch window $[e_l, l_l]$ $(e_l, l_l \leq T)$, find a set of time-feasible directed cycles covering the lanes in L of minimum total duration. Note that TCLCP is NP-Hard by Theorem 2.1 of Section 2.3.

A directed cycle C is time-feasible if we can identify a first arc in the cycle and we can assign a departure time d_a to the origin of each arc $a \in C$ in such a way that $d_a + t_a \leq d_{suc(a)}$ for all arcs in the cycle except for the last arc, where $suc(a)$ denotes the immediate successor of arc a in the cycle, and $e_l \leq d_a \leq l_l$ or $e_l + T \leq d_a \leq l_l + T$ if arc a is used to cover lane l, and such that the duration of the cycle is less than or equal to T, i.e., the time elapsed between the departure at the origin of the first arc and the return to the origin of the first arc is less than or equal to T. The latter condition ensures that the cycle can be repeated every period.

Observe that if a cycle is time-feasible, then it is time-feasible regardless of the arc chosen as the first arc. To see this, suppose there exists an arc and a departure time at the origin of that arc that results in a duration of the cycle of less than or equal to T. Since a single driver can complete the entire cycle in a single period, that driver can serve the same cycle every period. The driver visits each of the points (origins and destinations of the arcs) once in every period. In other words, the driver returns to each of the points of the cycle within at most time T from the departure from that point. But that means, that the cycle is time-feasible regardless of the arc chosen as first arc.

2.4.1 Solution Approaches

We focus on developing an effective and efficient heuristic for TCLCP as instances encountered in practice are expected to be large.

We have implemented a greedy heuristic that generates a large number of time-feasible cycles (potentially all) and greedily select a subset of those cycles to cover the lanes based on some criterion measuring the desirability or attractiveness of a cycle. After all lanes are covered we perform a local search step to improve the solution.

Note that a lane may be covered by more than one cycle, but that in all but one cycle the corresponding arc of the cycle represents asset repositioning (or deadheading). Consequently, the desirability or attractiveness of a cycle may change during the execution of the selection phase of the heuristic. By explicitly defining, in advance, the role of an arc in a cycle, i.e., whether it covers a lane or whether it represents repositioning, we can get around this issue at the expense of a larger set of cycles. For each cycle, we can generate its "siblings" by replacing one or more arcs covering lanes with arcs representing repositioning, ensuring that we never replace two consecutive arcs. The process is illustrated in Figure 2.2. If the cycle in Figure 2.2(a) represents the original cycle, then the cycles in Figure 2.2(b) represent its siblings (dashed lines represent asset repositioning).

The desirability or attractiveness of each cycle can now be established upfront, because we know precisely which arcs are used to cover lanes and which arcs are used for repositioning.

A natural way to capture the desirability of a cycle is through the ratio of the sum of the travel times of the lanes covered by the cycle and the duration of the cycle (the sum of travel and waiting time). This *cover ratio* takes on values in (0,1] and a higher value indicates a more desirable cycle.

The basic greedy heuristic iteratively selects a cycle with the highest cover ratio until all lanes have been covered. This involves sorting the set of cycles and, after selecting a cycle, deleting all cycles that cover one or more lanes of the selected cycle. These operations can efficiently be implemented using heaps and reference lists.

2.4.1.1 Cycle Generation

We generate time-feasible cycles using a recursive procedure. For each lane $\ell \in L$, we construct all simple paths starting with ℓ and ending with a lane arc and consisting of lane and repositioning arcs. We then connect the endpoints of the constructed path, if necessary, with the appropriate repositioning arc to convert the path into a cycle. To control the number of cycles generated, we restrict the travel time of the repositioning arcs on the path to be smaller than a predefined value R. (There is no restriction on the travel time of the repositioning arc connecting the endpoints of the path.) Sometimes, we will also limit the number of cycles generated by restricting the number of lanes in a cycle. Each recursion starts with a *base path*, initially just the lane arc ℓ. Then, to enforce the limit on the travel time of the repositioning arc, we find the lane arcs with an origin at travel time less than or equal to R from the destination of the last lane arc on the base path. For all such lane arcs $\hat{\ell}$, we check if the cycle consisting of the base path, a repositioning arc (if any) to the origin of lane arc $\hat{\ell}$, the lane arc $\hat{\ell}$, and the return repositioning arc (if any) to the origin of the first lane arc of base path is time-feasible. If so, we add the cycle to the list of generated time-feasible cycles and invoke a new recursion where the base path is the newly generated cycle without the final repositioning arc, i.e., lane arc $\hat{\ell}$ is the last arc on the extended base path.

For a cycle to be time-feasible, it must respect the dispatch windows and its duration has to be less than the period length. Because of the dispatch windows, the duration of a cycle may include waiting time. The waiting time on a cycle depends on the lane arc chosen to be the first arc of the cycle and the departure times at each of the lane arcs in the cycle. Therefore, the first arc of the cycle and the departure times at each of the lane arcs in the cycle should be chosen in such a way that the total waiting time along the cycle is minimum. Note that the set of departure times resulting in the minimum total waiting along a cycle is not unique. We set the departure time at the origin of the first arc to be the earliest time which results in the minimum total waiting time along the cycle and the departure times of the remaining lane arcs in the cycle to the earliest feasible departure time. This set of departure times yields the earliest departure time at the last lane arc in the cycle given that the waiting time along the cycle is minimum. Throughout the construction of time-feasible cycles we monitor and update, if necessary, the departure times on the active path to maintain this (invariant) property.

There may be flexibility with respect to the departure time at the origin of the first arc in the path, in the sense that departing a little later may not affect the total waiting time along the path. We define the difference between the latest and the earliest departure time at the origin of the first arc in the path resulting in minimum total waiting time along the path as the flexibility of the path denoted ϕ_P. Figure 2.4 illustrates the flexibility of a path with two lane arcs.

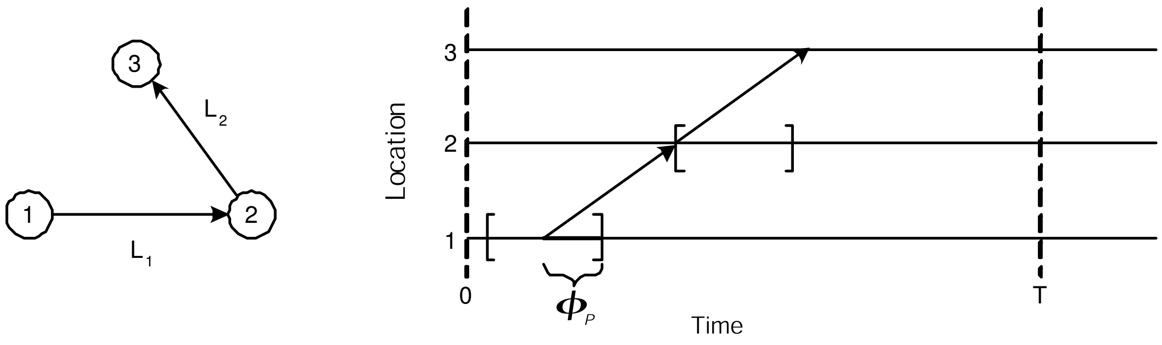


Figure 2.4: Flexibility of a path

Let σ_p denote the departure time at the first node of a path p and let τ_p denote the time required to traverse a path p. Let $\hat{\ell}$ be a candidate lane arc for expanding the base path P and let $\delta_{\hat{\ell}}$ denote the width of the dispatch window of lane $\hat{\ell}$. Furthermore, let $\bar{P}$ denote the base path P extended with the repositioning arc required to reach the origin of lane arc $\hat{\ell}$, i.e., $\bar{P} = P \cup (\text{head}(P), \text{tail}(\hat{\ell})$. The required time to traverse the cycle that forms when $\hat{\ell}$ is added to the base path is computed as follows. First, we compute the beginning and end of the dispatch window of $\hat{\ell}$ relative to σ_P, say t_1 and t_2 respectively. That is $t_2 = l_{\hat{\ell}} - \sigma_P$ if $\sigma_P + \tau_P \leq l_{\hat{\ell}}$ or $l_{\hat{\ell}} + T - \sigma_P$ if $\sigma_P + \tau_P > l_{\hat{\ell}}$ and $t_1 = t_2 - \delta_{\hat{\ell}}$. Then, the departure time of $\hat{ell}$ relative to σ_P, $d_{\hat{\ell}}$, is equal to $\max\{\tau_{\bar{P}}, t_1\}$. If t_1 is strictly greater than $\tau_{\bar{P}}$, then there will *potentially* be a waiting time at $\hat{\ell}$. Let $w_{\hat{\ell}}$ denote this potential waiting. However, some or all of this waiting time may be eliminated by departing later at the origin of the first lane arc of the base path. The departure time at the origin of the first lane arc of the base path can be delayed by at most ϕ_P, the flexibility of the base path. So the minimum waiting time at lane $\hat{\ell}$ is $\bar{w}_{\hat{\ell}} = \max\{w_{\hat{\ell}} - \phi_P, 0\}$. Note that the waiting time along the base path does not change and that the waiting time along the new path P_{new} ($P_{new} = \bar{P} \cup \hat{\ell}$) is equal to the waiting time along the base path plus $\bar{w}_{\hat{\ell}}$. We now have all the information we need to compute the duration of new cycle, and, if necessary, the duration, departure, and flexibility of the extended base path:

$$\begin{aligned}
\tau_{P_{new}} &= \tau_{\bar{P}} + \bar{w}_{\hat{\ell}} + t_{\hat{\ell}} \\
\tau_{C_{new}} &= \tau_{P_{new}} + t_{(head(P_{new}),tail(P_{new}))} \\
\sigma_{P_{new}} &= \sigma_P + (w_{\hat{\ell}} - \bar{w}_{\hat{\ell}}) \\
\phi_{P_{new}} &= \min\{\phi_P - (w_{\hat{\ell}} - \bar{w}_{\hat{\ell}}), t_2 - d_{\hat{\ell}}\}.
\end{aligned}$$

The above arguments show that we can handle lane dispatch windows during cycle generation without increasing the time complexity, because testing feasibility and maintaining path information takes constant time.

2.4.1.2 *Cycle Duration Minimization*

Recall that if a cycle is time-feasible, then it is time-feasible regardless of the lane arc chosen as the first arc of the cycle. Therefore, the first arc of a time-feasible cycle should be chosen such that the cycle has minimum duration. This is equivalent to choosing the first arc of a time-feasible cycle in such a way that the difference between the return to the origin of the first arc and the departure from the origin of the first arc in the next period is the maximum possible. The origin of the first arc of a cycle may be used as the domicile of the driver serving that cycle. By choosing the arc that results in minimum cycle duration as the first arc, we minimize the time the driver and the truck spends in waiting away from his domicile. Thus, we maximize asset utilization within the cycle. The cycle generation procedure described above does not necessarily result in an ordering of the arcs that results in the minimum cycle duration. To illustrate this, consider the instance shown in Figure 2.5. The two lanes $L_1 = (1, 2)$ and $L_2 = (2, 3)$ have dispatch windows $[e_1, l_1]$ and $[e_2, l_2]$ $(0 < e_1 < l_1 < e_2 < l_2 < T)$, respectively. The cycle generation procedure would produce the time-feasible cycle $(L_1, L_2, (3, 1))$ with a departure time at the origin of L_1 at l_1. However, it is easy to see that a shorter duration is obtained if L_2 is chosen as the first arc of the cycle with a departure time at the origin of L_2 at e_2, see Figure 2.6.

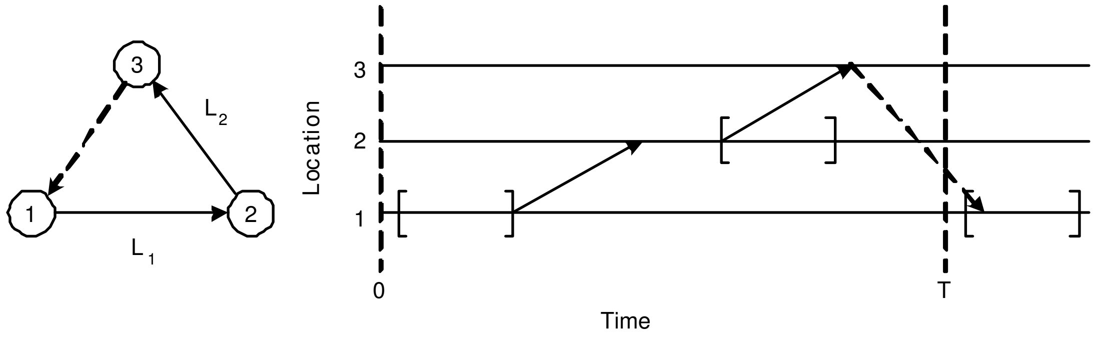


Figure 2.5: Constructed cycle with lane L_1 as first lane arc

Consequently, for each generated time-feasible cycle, we have to determine the lane arc that results in the minimum cycle duration when chosen as the first arc of the cycle. A relatively simple $O(n^2)$ algorithm, where n is the number of lane arcs in the cycle, simply

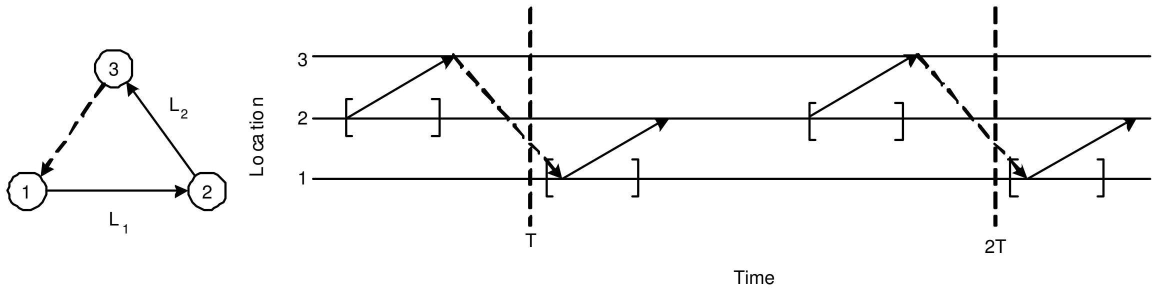


Figure 2.6: Constructed cycle with lane L_2 as first lane arc

traverses the cycle starting from each lane arc in the cycle, computes the duration, and in the end selects the lane arc that resulted in the minimum duration. A more efficient *linear time* algorithm exists, which is presented below. As the number of time-feasible cycles generated may be large, an efficient algorithm to identify the lane arc that results in the minimum cycle duration when chosen as the first arc is of utmost importance.

The linear-time algorithm first determines the earliest possible return time at a lane arc and then determines the latest possible departure at that lane arc in the next period ensuring minimum waiting time along the cycle. Consider time-feasible cycle $C = (1, 2, ..., k, 1)$, i.e., cycle C consists of k lane arcs labelled, without loss of generality, $1, 2, ..., k$. The cycle generation procedure has already provided the duration with lane 1 as the first arc of the cycle. Let C^i represent the same cycle, but with lane i chosen as the first arc, i.e., $(i, i+1, ..., k, 1, 2, ..., i-1, i)$. The algorithm makes one forward and one backward pass through the cycle. In the forward pass, we determine for each lane $i \in \{2, ..., k\}$ the earliest arrival time a_i^i at the origin of i and the associated departure time d_1^i at the origin of 1 that ensures minimum waiting time along the path $(1, ..., i)$, see Algorithm 2.5. In the backward pass, we determine for each lane $i \in \{2, ..., k\}$ the latest departure time d_i^i at the origin of i and the associated arrival time a_1^i at the origin of 1 ensuring minimum waiting time along the path $(i, ..., k, 1)$, see Algorithm 2.6. The difference Δ_i between departure time and arrival (return) time at i can now be computed as:

$$\Delta_i = d_i^i - a_i^i - (a_1^i - (d_1^i + T)).$$

The last term ensures that we arrive at the origin of 1 at or before the time of the departure

Algorithm 2.5 Forward Pass

```
$d \leftarrow e_1$
$\bar{s} \leftarrow l_1 - e_1$
$s \leftarrow 0$
**for** $i = 2, ..., k$ **do**
    $d \leftarrow d + t_{i-1,i}$
    **if** $d < e_i$ **then**
        $s_i \leftarrow min(e_i - d, \bar{s})$
        $\bar{s} \leftarrow min(l_i - e_i, \bar{s} - s_i)$
        $s \leftarrow s + s_i$
        $d \leftarrow e_i$
    **end if**
    $d_1^i = e_1 + s$
    $a_i^i = d + s_i$
**end for**
```

Algorithm 2.6 Backward Pass

```
$d \leftarrow l_1 + T$
$s \leftarrow 0$
**for** $i = k, ..., 2$ **do**
    $d \leftarrow d - t_{i-1,i}$
    **if** $d > l_i$ **then**
        $s_i \leftarrow (d - l_i)$
        $s \leftarrow s + s_i$
        $d \leftarrow l_i$
    **end if**
    $d_i^i = d$
    $a_1^i = l_1 + T - s$
**end for**
```

in the next period. The duration of C^i is equal to $T - \Delta_i$, therefore the lane that results in the largest value of Δ_i should be chosen as the first arc of the cycle.

Because the algorithm makes two passes through the cycle it has a linear time complexity.

2.4.1.3 Local Improvement

By adjusting the limit on the travel time of the repositioning arcs on the base path and the number of lanes in a cycle, we have control over the number of cycles produced during cycle generation. With a smaller travel time limit and number of lanes fewer cycles are generated. However, with fewer cycles to choose from, greedy selection may not produce high quality lane covers.

Therefore, we have developed a local improvement scheme that merges cycles from a cycle cover by removing the longest repositioning arcs from each cycle and by optimally reconnecting the two resulting directed paths to form another cycle. An example of a cycle merge is shown in Figure 2.3.

In this example, each of the two cycles consists of two lane arcs plus a single repositioning arc. In the merged cycle, the two repositing arcs have been deleted and replaced by two shorter repositing arcs.

Such a merge improves the current cycle cover if the sum of the durations of the initial cycles C_1 and C_2 is more than the duration of the merged cycle C_{12}. Given an initial cycle cover $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$, the optimal set of cycle merges can be obtained by finding a minimum cost matching on the digraph $D_{\mathcal{C}} = (V_{\mathcal{C}}, A_{\mathcal{C}})$ where $V_{\mathcal{C}} = \{1, 2, ..., N\}$, $A_{\mathcal{C}} = \{(i,j) : C_{ij}$ is time-feasible and $\tau_{C_i} + \tau_{C_i} > \tau_{C_{ij}}\}$ with arc weights $w_{ij} = \tau_{C_{ij}} - (\tau_{C_i} + \tau_{C_i})$ for $(i,j) \in A_{\mathcal{C}}$. Unfortunately, finding an optimal matching on $D_{\mathcal{C}}$ becomes computationally expensive when $D_{\mathcal{C}}$ gets large. Hence, we have also developed a *greedy merge heuristic.*

Our greedy merge heuristic identifies all feasible and beneficial cycle merges, finds the most improving cycle merge, implements that cycle merge, sets the resulting cycle aside, and then repeats. (See Algorithm 2.7 for a detailed description.)

Algorithm 2.7 GreedyMergeHeuristic($\mathcal{C}$)

```
for all cycles $C_i$ in $\mathcal{C}$ do
  if $C_{ij}$ is time-feasible and $\tau_{C_{ij}} < \tau_{C_i} + \tau_{C_j}$ then
    $Insert((i,j), \tau_{C_{ij}} - (\tau_{C_i} + \tau_{C_i}), ImprovementHeap)$
  end if
end for
while $ImprovementHeap \neq \emptyset$ do
  $(k,l) := Pop(ImprovementHeap)$
  if cycles $C_k$ and $C_l$ are not marked as merged then
    merge $C_k$ with $C_l$ and mark them as merged
  end if
end while
```

2.4.2 Computational Experiments

We have conducted a set of computational experiments to assess the overall effectiveness of the algorithms proposed and to analyze the impact of instance size (number of points and number of lanes) and instance characteristics (presence of clusters, supply chain structure, and dispatch time window width) on the performance of the proposed algorithms in terms of quality and efficiency.

We have evaluated the algorithms on randomly generated Euclidean instances. Random instance generation is controlled by the following parameters: the number of points, the fraction of points in clusters, the number of points per cluster, and the number of lanes. Clusters are introduced to represent geographical concentrations of points, such as metropolitan areas. For a set of instances we also impose a supply chain structure by dividing the set of points into three classes representing: suppliers, plants and distribution centers, and customers. The fraction of points belonging to each class and the fraction of lanes between points in any two classes are given. These instances will be referred to as supply chain instances (SC-instances). Given the number of points, the fraction of points in clusters, and the number points per cluster, the number of clusters and a cluster radius are determined. As the number of points in an instance increases, we let the cluster radius decrease in order to avoid overlapping clusters. Next, the centers of the clusters are determined uniformly at random within a $1,800 \times 1,800$ miles square region. The points within each cluster are generated by randomly determining their coordinates using a Normal distribution, $N(0,1)$, and the cluster radius. Finally, the points that are not in clusters are generated by determining their coordinates uniformly at random within the square region. If the instance being generated is a SC-instance, then each point generated is assigned to type: suppliers, plants and distribution centers, or customers with probability 0.2, 0.1, or 0.7, respectively. Once all the points are generated the complete bi-directed Euclidean graph defined by these points is formed. Next, the desired number of lanes are randomly selected from among the arcs of the complete bi-directed graph ensuring that each point has at least one lane incident to it and there are no lanes with an origin and a destination in the same cluster. For the SC-instances, we only allow lanes between suppliers and plants and distribution centers, among plants and distribution centers, and between plants and distribution centers and customers. Furthermore, we ensure that each supplier has at least one outgoing lane, each plant and distribution center has at least one incoming and one outgoing lane, and each customer has at least one incoming lane. Finally, after all the lanes are generated we determine the dispatch time window for each lane. Given a lane, first a day of the week is chosen randomly as the dispatch day. Once the dispatch day is set, the start time and width of the dispatch window are determined.

To properly analyze the performance of the algorithms, we have generated instances for several different parameter settings. We have varied the number of points, the fraction of points in clusters, the number of points per cluster, the number of lanes, and the width of the dispatch windows. More specifically, we generated instances with 300, 400, and 500 points, with a fraction of points in clusters of 0.5, 0.6, 0.7, and 0.8, and with an average number of 10 customers per cluster. The density of the instance is controlled by the lane-to-point ratio (ratio of the number of lanes and the number of points) with values 2 and 5.

2.4.2.1 Analysis of Algorithm Performance

With the first experiment, we aim to better understand the impact of the algorithm parameters that affect the construction of an initial solution, i.e., the generation and selection of time feasible cycles. Recall that to control the number of cycles generated (to reduce memory usage and computing time), we restrict the travel time of all but one of the repositioning arcs on the cycle to be smaller than a predefined value R. Table 2.10 presents the results for $R = 1, 10, 25$, and 50 miles. We limit the number of lanes in a cycle to at most six. Initial computational experiments revealed that cycles with a large number of lanes are rarely selected and limiting the number of lanes in a cycle, allows us to solve large scale problems without running into memory problems. The value of a solution, i.e., the sum of the durations of the cycles selected, is presented as a percentage over the lower bound obtained by solving the unconstrained lane covering problem (LCP). In this first experiment, we have used 48 randomly generated instances grouped into 6 sets based on their size (i.e., number of points and number of lanes). In each set, there are instances with and without a supply chain structure and with different cluster characteristics. Furthermore, we assume that all lanes have a 12 hour dispatch window which starts at 8am in the morning. The number of cycles generated and the CPU times required for constructing the solutions are presented in Table 2.11 and Table 2.12, respectively

Table 2.10: Percent deviation from the lower bound for the greedy construction heuristic.

#points	#lanes	R = 1	R = 10	R = 25	R = 50
300	600	51.90	41.55	28.96	25.55
300	1500	47.08	34.83	23.00	20.18
400	800	53.88	40.20	28.23	25.17
400	2000	48.14	33.13	22.45	19.91
500	1000	54.97	39.94	28.63	25.92
500	2500	49.79	32.73	22.86	20.65

Table 2.11: Number of cycles created by the greedy construction heuristic.

#points	#lanes	R = 1	R = 10	R = 25	R = 50
300	600	1,163	2,411	14,086	33,987
300	1500	5,501	29,754	733,299	2,946,147
400	800	1,578	4,047	24,882	50,374
400	2000	7,379	54,609	1,252,090	3,487,056
500	1000	1,996	6,730	57,276	128,566
500	2500	9,473	117,403	2,479,371	5,334,276

Table 2.12: CPU times in seconds for the greedy construction heuristic.

#points	#lanes	R = 1	R = 10	R = 25	R = 50
300	600	0.01	0.03	0.23	0.68
300	1500	0.05	0.44	16.86	79.70
400	800	0.01	0.04	0.44	1.06
400	2000	0.07	0.88	30.46	95.53
500	1000	0.01	0.08	1.20	3.11
500	2500	0.09	2.19	66.32	155.97

We observe that the solution quality increases steadily as the value of R increases, but that, at the same time, the number of cycles created and the run times increase exponentially. We also observe that the gap between the solution value and the lower bound value is about 5% smaller for instances with the higher lane-to-point ratio 5. This is most likely the result of the fact the number of opportunities for time-feasible continuous moves increases when the number of incoming and outgoing arcs at a point increases. Finally, we point out that searching for continuous moves can have substantial benefits as the non-collaborative solutions are significantly worse.

Next, we analyze the effect of limiting the number of lanes in a cycle during the cycle generation phase to at most K, for $K = 3, 4, 5$, and 6. We restrict the travel time of all but one of the repositioning arcs on the cycle to be smaller than 25, i.e., $R = 25$, and use the same randomly generated instances as before with 12 hour dispatch windows starting at 8am. The results can be found in Table 2.13.

Table 2.13: Percentage deviations from the lower bound for the construction heuristic with varying limits on the number of lanes in a cycle.

#points	#lanes	K = 3	K = 4	K = 5	K = 6
300	600	28.94	28.94	28.96	28.96
300	1500	22.83	22.94	23.00	23.00
400	800	28.37	28.25	28.23	28.23
400	2000	22.28	22.42	22.47	22.45
500	1000	28.54	28.59	28.61	28.63
500	2500	22.34	22.52	22.52	22.86

We see that the quality of the solutions is almost identical for all values of K. This suggests that high quality solutions typically involve only a small number of lanes. This may be a consequence of the chosen criterion for choosing cycles, i.e., the cover ratio. The cover ratio, i.e., the sum of the travel times of the lanes covered by a cycle and the duration of that cycle, favors cycles with little empty repositioning and waiting time. It is more likely to find cycles with a high cover ratio among cycles with only a few lanes. Therefore, the results may be different when other criteria are used for cycle selection. On the other hand, cycles with only a few lanes are preferred in practice as the chance for failed continuous moves is smaller in cycles with only a few lanes.

Next, we investigate the effectiveness of local search and the impact of using the greedy merge heuristic (GMH) as opposed to solving the matching problems optimally (OM). Our computational experiments revealed that the initial solution characteristics did not change the comparative performance of GMH and OM. Therefore, we compare the performance of the two methods starting from an initial solution consisting of only the trivial 2-cycles. We use the same instances as before. Note that local search is applied iteratively. After a set of cycle merges has been identified and implemented, we continue the search for improved solutions by setting up and solving another matching problem with time-feasible and beneficial cycle merges. The result can be found in Table 2.14, where we present the value of the solution obtained by the greedy merge heuristic (GMH) and the value obtained when the matching problem is solved to optimally (OM). For completeness sake and anticipating its use in later computational experiments, we also present the value of the non-collaborative solution. The value of the non-collaborative solution is calculated by assuming that each lane is covered by a simple 2-cycle containing the lane arc and its reversal.

Table 2.14: Percentage deviations from the lower bound and CPU times in seconds for the two local improvement heuristics.

		% deviation	% deviation		CPU seconds	
#points	#lanes	2-cycle	GMH	OM	GMH	OM
300	600	78.39	15.11	13.10	0.10	1.40
300	1500	77.83	10.01	8.09	0.58	11.18
400	800	80.24	14.76	12.79	0.17	2.60
400	2000	79.30	9.76	7.91	1.03	22.82
500	1000	81.43	14.58	12.68	0.26	4.57
500	2500	80.96	9.60	7.79	1.65	36.53

First, we observe that local improvement is very effective. Starting from solutions with deviations close to 80% from the lower bound, solutions with less than 15% deviations are obtained. Second, we see that the percentage deviations of the solutions obtained by the optimal matching are about 2% less than those obtained by the greedy merge. However, solving the matchings to optimality substantially increases the running time, particularly for the larger instances. As in the previous experiment, we observe that for the dense instances, with lane-to-point ratio 5, the solutions obtained are closer to the lower bound, but that the running time is an order of magnitude larger. This is due to the fact that the number of time-feasible and beneficial cycle merges is significantly larger. In the rest of the section, we use the greedy merge heuristic as our local improvement algorithm due to its computational efficiency.

The difference between the value of the solutions obtained and the value of the solutions to the unconstrained lane covering problem (the lower bound) can be attributed to two factors: (1) dispatch windows on the lanes, which render certain continuous moves infeasible, and (2) solving the problem heuristically. To get a sense of what portion of the difference can be attributed to the presence of dispatch windows, we first ran the greedy merge heuristic starting from the trivial 2-cycle cover ignoring any dispatch windows and then starting from the trivial 2-cycle cover respecting the dispatch windows. The results can be found in Table 2.15.

Table 2.15: Percent deviation from the lower bound with and without taking dispatch windows into account.

#points	#lanes	2-cycle solution without dispatch windows	2-cycle solution with dispatch windows
300	600	7.60	15.11
300	1500	5.24	10.01
400	800	7.20	14.76
400	2000	4.93	9.76
500	1000	7.70	14.58
500	2500	4.65	9.60

We see that about half of the difference can be attributed to the presence of dispatch windows whereas the other half is due to the use of a heuristic solution approach.

2.4.2.2 Analysis of the Value of Collaboration

The next sets of experiments are aimed at assessing the value of collaboration and the impact of dispatch windows on the opportunities for collaboration. In Table 2.16, we present the percentage deviations from the lower bound for the values of the solutions produced by the greedy heuristic followed by the greedy merge heuristic (for different values of R) and the values of the solutions produced by the greedy merge heuristic when started from 2-cycle solutions.

Table 2.16: Percentage deviations from the lower bound for the construction and improvement heuristics combined.

		non-collaborative	collaborative				
#points	#lanes		R = 1	R = 10	R = 25	R = 50	2-cycle
300	600	78.39	21.25	20.91	18.41	17.52	15.11
300	1500	77.83	14.52	13.79	12.24	11.74	10.01
400	800	80.24	20.99	20.09	18.08	17.38	14.76
400	2000	79.30	14.46	13.24	11.67	11.31	9.76
500	1000	81.43	20.87	20.05	17.90	17.38	14.58
500	2500	80.96	14.49	13.14	11.49	11.18	9.60

We can make the following observations regarding the results in Table 2.16. First and foremost that collaboration significantly reduces empty repositioning. Furthermore, we see again that local search is very effective. The solutions obtained for $R = 1, 10, 25$, and 50, are substantially better than those reported in Table 2.10. An additional 7-8 percent is shaved off from even the best solutions obtained using the greedy construction heuristic by itself. This indicates that to obtain high quality continuous move tours it is necessary to include several relatively long asset reposition moves. This is a valuable observation, because planners and analysts without access to decision support technology may not naturally consider such solutions. The presence of dispatch windows most likely increases the need to include relatively long asset repositioning moves in high quality continuous move tours as dispatch windows may invalidate connections that appear to be attractive geographically. Second, that the quality of the starting solution does not appear to have a major impact. Even starting from a trivial 2-cycle solution results in high quality solution. In fact, the best solutions are obtained when starting from a 2-cycle solution. As before the deviations from the lower bounds are about 5% smaller for the dense instances. The increase in running times as a result of local search is negligible (at most 2 seconds).

Next, we analyze the effect of the width of the dispatch windows on the number of time feasible cycles and the solution quality. For these experiments, we start the greedy merge heuristic from an initial solution generated by the greedy heuristic with $R = 25$ and from the trivial 2-cycle covers. We use the same randomly generated instances as before except for the dispatch window widths. We use dispatch window widths of 2, 4, 6, and 12 hours and choose the start of the dispatch windows randomly for instances with dispatch window widths less than 12 hours. Once chosen, the start of the dispatch windows for each lane is kept constant for all instances. The results are presented in Tables 2.17 and 2.18.

Table 2.17: Percentage deviations from the lower bound for the construction and improvement heuristics combined, with varying dispatch window widths.

		width = 2hrs		width = 4hrs		width = 6hrs		width = 12hrs	
#points	#lanes	GH	2-cycle	GH	2-cycle	GH	2-cycle	GH	2-cycle
300	600	31.45	26.16	27.38	22.27	24.13	19.47	18.41	15.11
300	1500	22.19	18.00	18.61	14.82	16.15	12.79	12.24	10.01
400	800	30.73	25.53	26.33	21.68	23.55	18.98	18.08	14.76
400	2000	21.40	17.57	17.86	14.50	15.49	12.51	11.67	9.76
500	1000	30.30	25.03	26.30	21.11	23.27	18.54	17.90	14.58
500	2500	21.30	17.36	17.67	14.22	15.09	12.24	11.49	9.60

Table 2.18: Number of cycles generated by the construction heuristic with varying time window widths.

#points	#lanes	width = 2hrs	width = 4hrs	width = 6hrs	width = 12hrs
300	600	3,917	5,256	6,780	14,086
300	1500	84,493	138,524	218,569	733,299
400	800	6,513	8,788	11,641	24,882
400	2000	145,007	240,355	376,604	1,252,090
500	1000	10,863	15,733	22,254	57,276
500	2500	289,811	513,722	857,076	2,479,371

As expected, we observe that as the dispatch window widths get narrower the quality of the solutions decreases substantially. In fact, the percentage deviations of the solutions from the lower bound with 2 hour dispatch windows are about 10% higher for low density instances and about 8% higher for high density instances than those with 12 hour dispatch windows. We see again that the impact of changes, in this case dispatch window width, is less severe for high density instances where the number of opportunities for beneficial continuous moves is larger. We also observe that as the dispatch window width increases the number of cycles generated by the greedy heuristic increases and so does the number of cycles that can be merged (not presented in the tables). Finally, we note that for all dispatch window widths starting the local improvement from the 2-cycle solution, as opposed to starting from the solution generated by the greedy algorithm, performs best.

Next, we analyze the effects of geographical instance characteristics on the value of collaboration. Instances are characterized by the number of points, the fraction of points in clusters, the number of points in a cluster, the number of lanes, and whether or not there is a supply chain structure. In Tables 2.19 and 2.20 we present results for instances grouped by whether or not a supply chain structure is present (y/n), their density (i.e., the ratio of number of lanes to number of points (2 or 5)), and the fraction of points in a cluster (0.5, 0.6, 0.7, or 0.8). Each of these 16 groups has three members with 300, 400, and 500 points. The dispatch window widths are set to 12 hours starting at 8am.

Table 2.19 shows that when a supply chain structure is present in instances, the solutions obtained are closer to the lower bound. We believe this suggests that when a supply chain structure is present an increase in reposition miles is inevitable, and hence the lower

Table 2.19: Effect of instance characteristics on the solution quality.

Group	SC	Density	Frac in Clst	R=1	R=10	R=25	R=50	2-cycle
1	Yes	2	0.5	15.23	15.90	15.23	14.92	12.65
2	Yes	2	0.6	14.94	15.42	14.89	14.52	12.15
3	Yes	2	0.7	14.67	15.39	14.81	14.14	11.55
4	Yes	2	0.8	14.66	15.29	14.40	13.86	11.77
5	Yes	5	0.5	8.23	9.23	9.14	9.18	7.60
6	Yes	5	0.6	8.28	9.25	9.30	9.30	7.49
7	Yes	5	0.7	8.04	9.02	8.85	8.72	7.31
8	Yes	5	0.8	7.83	9.25	9.16	9.03	7.11
9	No	2	0.5	28.21	26.60	22.61	21.58	18.07
10	No	2	0.6	27.26	25.58	22.68	21.59	18.01
11	No	2	0.7	26.22	23.96	20.26	19.67	17.09
12	No	2	0.8	27.11	24.67	20.17	19.15	17.23
13	No	5	0.5	21.13	18.67	15.96	15.37	12.73
14	No	5	0.6	20.86	17.60	14.80	14.08	12.30
15	No	5	0.7	20.96	17.28	13.93	13.17	12.07
16	No	5	0.8	20.58	16.82	13.02	12.30	11.73

bound is tighter. The reason for the increase in reposition miles is the fact that a large fraction of the points has no incoming arcs (points representing suppliers) and an even larger fraction of the points has no outgoing arcs (points representing customers). Therefore, fewer natural connections are present. This view is supported to some extend by the results in Table 2.20, which show that the number of cycles generated in the construction phase is smaller for instances with a supply chain structure. The number of cycles generated in the construction phase also suggests an explanation for the smaller gaps we have observed for high density instances. The presence of dispatch windows invalidates many continuous moves that look good from a geographical perspective, i.e., involve only a short repositioning move. Relatively speaking, this happens less frequently in high density instances because the in- and out-degree of the points are higher. This is reflected in the number of time-feasible cycles generated. When the density is high, the number of cycles generated is much higher. Furthermore, we need to keep in mind that the lower bound computation completely ignores dispatch windows. Combined these observations clarify the differences in gap sizes between low and high density instances.

Table 2.20: Effect of instance characteristics on number of cycles generated during the construction phase.

Group	SC	Density	Frac in Clst	R=1	R=10	R=25	R=50
1	Yes	2	0.5	1,596	3,602	12,974	23,811
2	Yes	2	0.6	1,551	4,097	28,656	57,194
3	Yes	2	0.7	1,610	4,901	34,226	69,579
4	Yes	2	0.8	1,602	5,932	77,522	193,433
5	Yes	5	0.5	4,901	29,301	368,195	1,041,817
6	Yes	5	0.6	4,729	37,888	1,100,413	2,742,038
7	Yes	5	0.7	4,937	57,204	1,418,979	4,404,726
8	Yes	5	0.8	4,953	92,221	1,974,056	4,926,831
9	No	2	0.5	1,540	3,250	13,819	25,873
10	No	2	0.6	1,568	3,915	21,916	50,602
11	No	2	0.7	1,573	4,367	27,645	60,103
12	No	2	0.8	1,592	5,103	39,895	87,211
13	No	5	0.5	9,591	47,051	563,984	1,533,128
14	No	5	0.6	10,118	70,509	1,273,310	4,046,833
15	No	5	0.7	10,060	90,686	1,983,492	5,783,194
16	No	5	0.8	10,314	113,183	3,167,453	7,046,333

2.4.3 A Real-World Case Study

Group purchasing organizations have become common place in the modern business world. A group purchasing organization seeks to achieve cost reductions by combining purchasing power to negotiate discounts. Their success in the area of product procurement, e.g., raw materials, has lead to companies to explore whether a group purchasing organization may also be effective when it comes to procuring services, e.g., truckload transportation.

To assess the potential value of collaborative purchasing of truckload transportation services and to prepare itself for negotiations with truckload carriers a group purchasing organization has to solve a time-constrained lane covering problem.

We have conducted a study for one such organization in which we assessed the potential value of collaborative transportation procurement for individual member companies, by coordinating purchasing of inbound and outbound truckload transportation services, and across member companies, by collaboratively purchasing all truckload transportation services.

To estimate the savings associated with offering collaborative continuous moves to a carrier, a carrier pricing model had to be developed. The carrier pricing model we developed takes as input a continuous move path (instead of a cycle) with an origin, a destination, the distance traveled between the origin and the destination, and the time elapsed between departing from the origin and arriving at the destination, and computes the associated charge. The charge is computed as the sum of a fixed cost component, based on the time elapsed between origin and destination, and a variable cost component, based on the distance traveled between origin and destination. The fixed cost per week, set at \$1,600, covers tractor and trailer depreciation and interest, driver compensation and fringes, insurance, tags, taxes, and licenses. The fixed cost of a path is computed by multiplying \$1,600 by the fraction of the week used by the path. The variable operating cost per mile, set at \$0.45, covers fuel and oil, tractor and trailer maintenance, and tractor and trailer tires.

In addition to the incurred direct costs, it is assumed that the carrier accounts for potential asset reposition to the next pickup location and for potential delay at the next pickup location. Asset repositioning and delay depend on many factors, such as the location of the destination, the time of arrival at this location, and the carriers demand patterns. For the study, asset reposition of a 100 miles and a delay of 8 hours are assumed, i.e., 100 miles are added to the distance traveled between origin and destination and 10 hours are added to the time elapsed between departing from the origin and arriving at the destination (8 hours plus two hours for traveling the additional 100 miles). Finally, the carrier is assumed to charge overhead and profit as a percentage of revenue, set at 15% and 10% respectively. Consequently, the formula for computing the charge for a path P, c_P, to a shipper is

$$c_p = \frac{4}{3}\left(1600\left(\frac{duration(P)+10}{168}\right) + 0.45\ (mileage(P)+100)\right). \tag{2.1}$$

Given the carrier pricing model, the savings associated with a collaborative continuous move path are estimated as the difference between the sum of the charges for the lanes on the path and the charge for the path. The savings represent (are the result of) elimination of some of the carrier's charges in anticipation of asset repositioning and delay.

In addition to using the carrier pricing model developed in the above section, to provide realistic estimates of time elapsed between departure at the origin and arrival at the destination, we properly account for the Hours of Service regulations (those in effect in 2004), i.e., a mandatory rest of 8 hours after driving for 10 hours or being on duty for 15 hours. As a result, the travel time between two locations is no longer a constant because it depends on the history of the driver. We also include a one hour rest after every four hours of driving. In addition, each location has a time window specifying a period of the day during which a pickup or delivery could take place. Furthermore, some locations require life loading whereas other locations used spot trailers (when spot trailers are used loading or unloading is fast as it only involves a change of trailers; when life loading is required no change of trailers takes place and a driver has to wait until loading or unloading is completed).

A typical all-member instance for a single week involves about 750 locations and 5500 lanes. The potential savings due to continuous moves were estimated to be in the order of 9-10%. (Due to confidentiality agreements, we are unable to provide more details.)

Since we are unable to present detailed results for the instances of the real-world case study, we have conducted a set of computational experiments on slightly simplified instances, in which we do not account for loading and unloading times, and where we use a version of the algorithm that ignores Hours of Service regulations.

In the first experiment, we investigate the potential savings and whether the obtained savings are sensitive to the algorithm settings for the construction phase (local search is applied in all variants tested). The results can be found in Table 2.21, where the savings are computed as $\frac{\text{one-way charges - continous move charges}}{\text{one-way charges}} \times 100$ percent.

Table 2.21: Savings from continuous moves.

Instance	#locations	#lanes	R = 1	R = 10	R = 25	R = 50	2-cycle
i1	103	1078	5.39%	5.36%	5.44%	5.34%	4.24%
i2	65	140	6.97%	7.28%	6.94%	6.99%	6.81%
i3	215	2346	9.66%	10.02%	10.04%	10.10%	8.81%
i4	404	2074	12.42%	12.20%	12.45%	12.71%	11.58%
i5	737	5445	11.87%	12.04%	12.02%	-	11.33%
i6	244	2889	10.47%	10.51%	10.65%	10.66%	9.58%
i7	427	2524	13.04%	12.92%	13.24%	13.58%	12.17%
i8	240	2828	10.23%	10.52%	10.73%	10.68%	9.24%
i9	215	2346	9.66%	10.02%	10.04%	10.10%	8.81%
i10	244	2889	10.47%	10.51%	10.65%	10.66%	9.58%
i11	240	2828	10.23%	10.52%	10.73%	10.68%	9.24%
i12	236	2352	8.69%	8.83%	8.90%	8.92%	7.93%
i13	130	1503	5.03%	5.31%	5.24%	5.28%	4.49%

We see that the savings range from about 5.5 percent to a little over 13 percent, where

Table 2.22: Savings from continuous moves for different dispatch window widths.

	width = 2hrs		width = 4hrs		width = 6hrs		width = 12hrs	
Instance	GH	2-cycle	GH	2-cycle	GH	2-cycle	GH	2-cycle
i1	4.15%	3.02%	4.57%	3.33%	4.89%	3.56%	5.44%	4.24%
i2	5.12%	4.74%	6.85%	6.15%	6.79%	6.69%	6.94%	6.81%
i3	7.61%	6.85%	8.50%	7.63%	8.96%	8.18%	10.04%	8.81%
i4	10.23%	9.77%	11.48%	10.77%	11.86%	11.21%	12.45%	11.58%
i5	9.42%	9.28%	10.51%	10.25%	11.26%	10.78%	12.02%	11.33%
i6	7.84%	7.44%	9.01%	8.27%	9.64%	8.93%	10.65%	9.58%
i7	10.77%	10.26%	11.86%	11.35%	12.63%	11.72%	13.24%	12.17%
i8	7.79%	7.09%	8.81%	7.99%	9.34%	8.61%	10.73%	9.24%
i9	7.61%	6.85%	8.50%	7.63%	8.96%	8.18%	10.04%	8.81%
i10	7.84%	7.44%	9.01%	8.27%	9.64%	8.93%	10.65%	9.58%
i11	7.79%	7.09%	8.81%	7.99%	9.34%	8.61%	10.73%	9.24%
i12	6.78%	6.05%	7.66%	6.89%	8.27%	7.36%	8.90%	7.93%
i13	4.27%	3.51%	4.75%	3.94%	5.08%	4.23%	5.24%	4.49%

the savings tend to be larger when the size of the instance is larger. It also appears that for these instances the quality of the initial tour has an impact on the quality of the final tour. The quality of the tour obtained after local search when starting from the trivial 2-cycles is universally worse than the quality of the tour obtained after local search when starting from a tour constructed with our greedy heuristic. One instance could not be solved for R=50 because the number of tours generated was too large to fit into memory.

Next, we investigate the impact of the size of the dispatch windows at the locations on the quality of the tour. The results when the initial solutions are obtained by the greedy construction heuristic with $R = 25$ and the trivial 2-cycles can be found in Table 2.22.

As expected, we see that the width of the dispatch windows has an impact on the savings that can result from the use of continuous moves. The smaller the dispatch window width the smaller the savings. This is due to the increased chance of waiting time between two consecutive moves. A large waiting time between two consecutive moves renders a continuous move path unprofitable.

Finally, we investigate the impact of a limit on the number of lanes in a continuous move path. In practice, the chance for breaking a continuous move, i.e., being unable to execute the complete sequence of moves as planned, increases when the number of lanes in a continuous move path is large. Therefore, in practice it is preferable to achieve the bulk of the savings from continuous move paths with a relatively small number of lanes. We limited the number of lanes allowed in a continuous move path to 3, 4, and 5 respectively. The results when the initial solutions are obtained by the greedy construction heuristic with $R = 25$ and the trivial 2-cycles can be found in Table 2.23.

Table 2.23: Savings from continuous moves for different limits on the number of lanes in the path.

	K=3		K=4		K=5	
Instance	GH	2-cycle	GH	2-cycle	GH	2-cycle
i1	5.38%	4.17%	5.44%	4.24%	5.45%	4.25%
i2	6.94%	6.74%	6.94%	6.81%	6.94%	6.81%
i3	9.74%	8.45%	10.04%	8.81%	10.13%	8.87%
i4	11.89%	10.52%	12.45%	11.58%	-	11.72%
i5	11.52%	10.34%	12.02%	11.33%	-	11.46%
i6	10.30%	9.21%	10.65%	9.58%	10.66%	9.65%
i7	12.66%	11.16%	13.24%	12.17%	-	12.27%
i8	10.33%	8.93%	10.73%	9.24%	10.62%	9.33%
i9	9.74%	8.45%	10.04%	8.81%	10.13%	8.87%
i10	10.30%	9.21%	10.65%	9.58%	10.66%	9.65%
i11	10.33%	8.93%	10.73%	9.24%	10.62%	9.33%
i12	8.69%	7.71%	8.90%	7.93%	8.96%	7.95%
i13	5.24%	4.41%	5.24%	4.49%	5.29%	4.49%

We see that limiting the number of lanes in a continuous move path has little impact on the savings. Allowing at most three lanes in a continuous move results in savings that are about 0.5 percent less than when at most four lanes are allowed. The decrease in savings is even smaller when we compare allowing at most four lanes to at most five lanes in a continuous move path, less than 0.1 percent (for all instances that could be solved).

CHAPTER III

BIDDING IN SIMULTANEOUS TRANSPORTATION PROCUREMENT AUCTIONS

3.1. Introduction

This chapter is devoted to the design and analysis of bidding strategies for a carrier participating in multiple simultaneous single-lane auctions. The challenge for the carrier is to determine bid prices, and thus potential revenues, while operating costs are uncertain as they depend on the lanes won, which is not known until the end of the auctions. In determining bid prices the carrier has to consider potential revenues, probable costs, and possible competitors' bids, which all have an impact on the carrier's profit. The focus is on incorporating anticipated bidding behavior of competitors when determining bid prices, as competitors' bids greatly affect the outcome of the auctions and therefore the set of lanes the carrier ends up winning, which in turn determines the carrier's operating cost. A bid price optimization problem with an objective of maximizing a carrier's expected profit is formulated and algorithms for its solution are designed and implemented. The value of bid price optimization for carriers is empirically evaluated on randomly generated instances simulating a real-life environment.

The contributions of the research presented in this chapter can be summarized as follows:

- It provides optimization based bidding (or dynamic pricing) algorithms for carriers participating in multiple simultaneous truckload procurement auctions. We are the first to study simultaneous truckload procurement auctions from a carrier's perspective. Most of the literature has focused on the design or the selection of auction mechanisms with desirable properties or on the characterization of equilibrium strategies for carriers. The literature that does study transportation procurement auctions from a carrier's perspective has focused on combinatorial or sequential auction settings.

- A stochastic bid price optimization problem, with the objective of maximizing a carrier's expected profit, has been formulated and an iterative coordinate search algorithm for its solution has been developed. The coordinate search algorithm produces optimal or close to optimal solutions for small to medium size practical instances in a short amount of time.

- A simulation study has been conducted that demonstrates the value of using bid price optimization. A carrier relying on the bid price optimization outperforms the competition, in terms of profit, especially in situations where exploiting network synergies is critical.

Before formally introducing the bid price optimization problem, we present an overview of transportation procurement and a review of relevant literature.

3.2. Transportation Procurement Overview

Traditionally, firms interested in shipping goods buy transportation services using requests for proposals (RFPs), leading to contract prices that are typically in effect for one to two years [68]. The traditional RFP process that most large shippers used until the late 1990s, and many still use today, typically begins by estimating the freight the shipper will need to ship in the coming year, usually by compiling the prior years movements [32]. Advertisements are placed in local or national media to transmit information to the carriers as a list of lanes on which the shippers expect the carriers to bid. Carriers then quote the prices at which they are willing to haul the loads. The shipper evaluates the bids lane by lane, using a single criterion, usually price, to select the winner. This process is nothing but a first-price sealed-bid auction. Since there are multiple lanes on auction, this process is in fact a simultaneous multiple-unit auction [44]. However, most shippers look at it as a set of individual auctions, one for each lane, ignoring lane interdependencies which play an important role in truckload (TL) carriers' operating costs.

The developments in information and communication technologies, especially widespread availability and use of the internet, facilitate quick, convenient, and cost-effective transactions between shippers and carriers. Many shippers form private transportation exchanges (or marketplaces) to procure transportation services where only a few carriers selected by the shipper compete for the lanes of the shipper. These private marketplaces enable shippers to work with a few trusted carriers, and thus are predominantly used for procurement of long term contracts. Alternatively, the shippers can procure transportation services using public exchanges such as www.freemarkets.com, where many shippers and many carriers are involved. Mostly, these public exchanges act as a spot market enabling the shippers to satisfy immediate and short term transportation needs resulting from fluctuations in shipment volume in a cost effective way. Regan and Nandiraju [59] provide a review of transportation marketplaces and their characteristics.

A carrier participating in procurement auctions has to take several factors into account when deciding on a bid for a given lane. Namely, (1) Type of auction mechanism in place, (2) Carrier's own cost structure, (3) Competitors' bidding strategies.

Note that the carrier's price must be high enough to make serving the shipper profitable but low enough to beat the competitors' prices. The revenue generated from an auctioned lane the carrier wins is determined by the type of auction run by the shipper. In a first-price sealed-bid auction, the carrier's revenue is equal to the carrier's own bid. In a second-price sealed-bid auction, the carrier's revenue is equal to the second highest bid, which the carrier does not know when he places his bid. In sealed-bid auctions, the carrier cannot change his bid once he places it, so the carrier must predict the competitors' bids and carefully place his bid. In descending procurement auctions, the bidders are allowed to decrease their bids. So, a carrier may start by placing a high bid and revise it as competitors' bids are revealed.

In order to assess the profitability of serving a customer request at a given price, the carrier must estimate the incremental cost of serving that customer request. A TL Carrier's operating costs is determined by his vehicle routing and fleet management technology and the interaction between the lanes to be served by that carrier. The set of lanes the carrier ends up serving is uncertain due to the participation of other carriers in the auction. The probability that a carrier wins a certain lane depends on the carrier's bid and the competitors' bids. In simultaneous transportation procurement auctions, the bidders are susceptible to the so called "exposure problem". That is, the carrier may fail to win one of the lanes which has strong synergies with the lanes he wins, leading to an unprofitable outcome for the carrier. In this setting, it is critical that the carrier incorporates his competitors' bidding strategies.

Recently, several large shippers and third-party-logistics (3PL) providers have turned to combinatorial auction mechanisms to procure transportation services. Combinatorial auctions enable the carriers to bid more aggressively without running into exposure problems by allowing conditional bids on bundles of lanes. When bidding on bundles, each carrier may submit multiple quotes for a single lane, because each lane can be a part of many packages. The shipper may therefore receive many more bids than the number of lanes, with overlapping lane quotes, making evaluating these quotes much more difficult than evaluating individual lane quotes. The shipper must solve the winner determination problem, which is a challenging combinatorial optimization problem [15]. Not every shipper has the sophisticated technology to run combinatorial auctions and it is natural that a carrier interacts with multiple shippers. Thus, bidding in multiple simultaneous procurement auctions is still a real challenge faced by carriers.

3.3. Literature Review

3.3.1 Auction Theory

Auctions have been used by people for thousands of years as market mechanisms. Greek historian Herodotus described the sale of women to be wives in Babylonia around the fifth century B.C. The auction of plundered bounty was common during the closing years of the Roman Empire. In China, the personal belongings of the deceased monks were auctioned off as early as the seventh century A.D. [17] [69]

Today, auctions account for an enormous amount of economic activity. Governments use auctions to sell treasury bills, radio frequency spectrums, foreign exchange, mineral rights including oil fields, and other assets such as firms to be privatized. Government contracts are usually awarded using procurement auctions. Firms seeking to buy inputs or subcontracting work also use procurement auctions. Houses, cars, antiques, artwork, agricultural produce and livestock are commonly sold through auctions. Internet auction web-sites such as Ebay and Yahoo are used to sell almost anything. Furthermore, economic transactions such as take over battles are auctions in effect if not in name.

Auctions are simple but useful and practical price discovery mechanisms to extract buyers' or sellers' valuations, especially when there is uncertainty about the value of an object or service. If there were no uncertainty, the seller/buyer would just transact with the buyer/seller that had the highest valuation. The term "auction" usually refers to the case that involves a seller and several buyers. Note that procurement auctions are of type "reverse" auctions because sellers (carriers) bid instead of buyers (shippers); prices are bid down instead of up. In "reverse auctions" sellers have a production or service costs, while in "forward auctions" buyers have a valuation of the object/service to be purchased. Most of the auctions studied in literature are forward auctions. On the other hand, models and intuition derived for most "auctions" can be easily reversed and applied to "reverse auctions" and vice versa [66].

Auction analysis is one of the most successful applications of game theory, because game theory formulations of auctions formally capture market competition and strategic interactions. Auctions are modeled as non-cooperative strategic games in which the players are the bidders and the seller(s). The bidders decide how much to bid, the seller decides the auction format and rules (the auction design). Payoffs depend on the design of the auction and the bids, but typically the surplus from the trade is split in some way between the winning bidder and the seller. Objectives of the game theoretic study of auctions are to understand: (1) how bidders decide how much to bid, (2) the consequences of the buyers bidding strategies for the expected price that the seller receives (3) the consequences of the sellers choices regarding auction design for the expected price that the seller receives (4) the circumstances in which auctions are an *efficient* mechanism (the item goes to the bidder who values it the most) for allocating scarce resources.

Single-item auctions are well understood, since they have been studied in economics as games of incomplete information for more than 40 years [81]. Several models for the auction of a discrete item with varying assumptions on the behavior of bidders have been studied. These models vary according to the amount of information available to each bidder, whether each bidder knows his own valuation for certain or bases his value somewhat on what others think, and the amount of affiliation or correlation among the values of bidders.

Auction theorists have studied in detail the four standard auction types: English (ascend the price until only a single bidder remains), Dutch (descend the price until the first bidder bids and wins the item), first-price sealed-bid (each bidder submits a price for the item in question unknown to the rest of the bidders, the winner pays the price he submitted), second-price sealed-bid (same as first-price sealed-bid except, the winner pays the second highest price). English (ascending) auction is shown to be both *incentive compatible* (truth telling is the best strategy) and *efficient*. The second-price sealed-bid auction displays incentive compatibility. Under an assumption that a bidders valuation does not depend on the valuation information revealed by her opponents, the second-price sealed-bid is strategically equivalent to the English auction. Auction theorists show that under mild assumptions, the Dutch and the first-price sealed-bid auction are strategically equivalent [44].

The conclusions reached in theoretical work regarding the standard auctions no longer hold when the simplifying assumptions is removed and the model is extended to describe the real world more accurately. For instance, the equivalence between the English and second-price auctions breaks down when bidders utility is governed by affiliated signals [56]. In this setting the English auction has higher expected revenue, because bidders in the English auction learn as the auction progresses and can bid more competitively with better information. On the other hand, lack of information about the value of the item induces conservative bidding in the second-price auction. The complete result for single-item auctions is that in terms of maximizing auctioneer revenue, the English auction outperforms the second-price auction, which in turn outperforms the first-price auction [44]. Also, a recent paper [16] shows that slow Dutch auctions (in which the price drops slowly over the course of a few days in an Internet auction) may produce more revenue than the first-price auction (see also the work by Lucking-Reiley [51]). Here, the value of time must be included as a transaction cost in models of Dutch auctions, and failure to capture all costs in an auction model may skew an auctioneer's decision-making criteria when selecting a particular auction.

The ranking of revenues across various auction mechanisms, given the problem of asymmetric bidders, has not yet been established. There are, however, a number of studies examining the ranking of revenues in the single unit case when bidders are asymmetric ex ante; the reader is referred to Klemperer [42] discussing the "wallet game" and Maskin and Riley [53] who compare revenues under a first-price sealed bid mechanism with those yielded by an English auction, assuming private values.

In recent years, the design and analysis of multi-item auctions with multi-item demand have received greater attention than single-item auctions. One stream of research, inspired by the U.S. spectrum auctions, has analyzed the case when items are heterogeneous [19, 55]. Some studies characterize equilibria in different auction mechanisms and compare revenue and efficiency when there exist synergies between the items for the bidders [34, 45, 9, 10, 11, 1, 43]. Assuming simple private value models, the studies show that a combinatorial auction is the most efficient mechanism, followed by the simultaneous multiple round auction. For a study of a multi-item auction with common values and synergies, see Rosenthal and Wang [65]. The simultaneous sealed-bid auction seems to be less efficient but it generates higher revenue than both the simultaneous multiple round and the combinatorial auction. The reason is that bidders bid more aggressively in the simultaneous sealed-bid auction than in the other mechanisms. The bidder submits bids on separate items that are above his valuation for each separate item, facing a possible loss, i.e. *exposure problem*, in case he wins only a fraction of the items. This risk of a negative profit is compensated by the bidder's potential gain in case of winning all items.

Combinatorial bidding can be used in both simultaneous one-shot (bidders may bid on any package of units they choose) and multiple round auctions (bidders may add new packages during the course of the auction). However, allowing for combinatorial bids may give rise to a free-rider (sometimes referred to as a threshold) problem, thereby introducing inefficient equilibrium outcomes that would be avoided in an auction without combinatorial bidding [55].

Due to the lack of theoretical results, laboratory experiments have been designed to test the revenue and efficiency ranking of multi-item auction mechanisms when bidders have synergies. It is demonstrated that a simultaneous multiple round (SMR) auction allowing for combinatorial bids dominates other mechanisms, both in terms of revenue and efficiency [49] [12]. In another work, it is reported that a SMR auction allowing combinatorial bids generates higher efficiency but lower revenues than a SMR auction without package bids, the reason being that bidders in the latter auction make losses because of failed contract aggregations [20]. In these lab experiments with combinatorial auctions, the occurrence of a free-rider was not observed. Further, real-life applications of procurement auctions where suppliers have had the option to submit combinatorial bids on bundles of contracts have generated substantial savings in procurement costs of transportation services [48] [15] [68] and wooden packaging material [78]. On the other hand, allowing combinatorial bids in procurement auctions may make it easier for a large firm to win a large number of contracts jeopardizing the competitiveness of the market. Data from a test of combinatorial procurement auctions for road markings in Sweden confirm this risk, especially if there are only a few large bidders [52].

In multi-item settings where bidders have diseconomy in the number of items that they are awarded, a few studies analyze bidding behavior in simultaneous auctions. When making multiple bids, bidders randomize over a set of available bids in a way that generates inefficient outcomes [26, 47]. In a simultaneous two-contract procurement auction model, where the designated winner is allowed to withdraw his bid, it is shown that if the incremental cost of winning an additional item is relatively high, it is more profitable to keep only one contract even if one were to win both [82]. Bidding then becomes more competitive on both contracts, lowering the procurer's cost. An analytical problem that arises in a sequential auction when bidders have interrelated values for units is that bidders' beliefs will become asymmetric during the course of auction even if beliefs are symmetric ex ante. After the first round, bidders update their beliefs about the values of the losing bidders and the value of the winning bidder, thereby creating the asymmetry. The sequence of second-price auctions result in an efficient allocation when two units are sold [41]. When more than two items are sold in a multi-item demand sequential auction (first-price or second-price auction) with diminishing valuations, the extent of inefficiencies created by the bidder asymmetries is unclear.

3.3.2 Procurement in Logistics & Transportation

Procurement auctions existed in the logistics and transportation industry in the form of Request for Proposals (RFPs) or Request for Quotations (RFQs). With the advent of internet technologies and their widespread adoption, internet emerged as a platform for facilitating business transactions such as buying and selling of goods, integrating supply chains, and interacting with partners and employees. Following the success stories of the early adopters, firms are investing billions of dollars in electronic procurement (e-procurement) technologies (e-markets) to achieve greater efficiency. See Elmaghraby [23] for a review of the current state of procurement in the industry.

The strategic significance and practical implications of using e-procurement attracted significant research efforts from the operations research and management science community. It is important to note that most researchers take the perspective of the buyer (or auctioneer), focusing on the design of new auction mechanisms or analyzing existing ones [4, 57] . To the best of the author's knowledge, relatively little research literature exists about bidding strategies in procurement auctions [85]. Particular attention has been paid to combinatorial procurement auctions which allow the suppliers (bidders) to express volume discounts, incompatible products/services, and complementary products/services. These auctions are preferred on the premise of allowing the buyer to benefit from increased price competition and the ability to specify product quality, delivery time, and other non-price attributes, while alleviating the strategic, logistical, and scaling problems of suppliers [4]. The buyer running the combinatorial auction must solve an $\mathcal{NP}$-Hard optimization problem to determine to which supplier each contract would be awarded. Elmaghraby and Keskinocak [24, 25] provide an overview of the state-of-the-art in combinatorial procurement auctions.

Because of inherent significance of synergies between shipping lanes for transportation carrier operations, the use of combinatorial auctions has been proposed for procurement of transportation services. Again, these auctions have mostly been analyzed from the shippers' perspective by researchers and practitioners alike. Moore et al. [58] describe the formulation of an optimization-based bidding approach for Reynolds Metals which was not fully implemented in practice due to limited computer capabilities available at the time. Ledyard et al. [48] describe the use of a “combined value” auction for purchasing transportation services for Sears Logistics Services in the early 1990s. Elmaghraby and Keskinocak [24] discuss how several companies, including Home Depot, Wal-Mart Stores, Compaq Computer Co., Staples Inc., the Limited Inc., and others, utilized optimization- based procurement tools to procure transportation services. Sheffi [68] provides a survey of the literature and practical results on the use of combinatorial auctions in procurement of truckload transportation services.

On the other hand, determining the optimal bid values for a carrier poses several challenging research questions. In essence, the problem that the carrier faces in this context is a pricing problem. Most of the literature on carrier operations treated pricing and cost minimization problems separately, and focused solely on minimizing the cost of serving the customers by solving fleet management and vehicle routing problems (VRP). See Crainic and Laporte [18] for a review of fleet management problems and Toth and Vigo [77] for a survey of VRP and its variants. Some of the past work considers the problem of how much the total profit would change when an additional load is introduced into a dynamic fleet management system (Powell [62, 63], Powell et al. [64], Topaloglu and Powell [76]). Although these works certainly are valuable, today’s market conditions call for a more direct and integrated approach, especially in the case of auctions. Recently, the study of transportation procurement auctions from the carrier perspective has received some attention. In combinatorial auctions, Song and Regan [70, 71] propose optimization based bid construction strategies for carriers. An et al. [3] propose a generic synergy model for constructing combinatorial bids. In these studies, the lane bundles are formed based on anticipated cost of serving each of the bundles and then the bid on each bundle is obtained by adding a fixed markup to this anticipated cost. They ignore the interactions of the carrier with other shippers and competing carriers.

Also receiving attention is bidding strategies in sequential transportation procurement auctions where multiple auctions take place in sequence, i.e. one at a time ruling out the possibility of having to bid on two or more auctions simultaneously. In this setting, the carrier must incorporate pricing decisions into dynamic fleet management decisions. Figliozzi [31] in his thesis studies the impact of different dynamic pricing and fleet management technologies on the revenues and operating costs of a carrier in a spot truckload procurement market. Mes et al. [54] consider a real-time dynamic pickup and delivery problem and propose a pricing strategy based on dynamic programming where the impact of a job insertion on future opportunities is taken into account. Douma et al. [21] consider a multi-company, less-than-truckload, dynamic VRP and use revenue management concepts to price the loads dynamically under capacity restrictions and taking into account possible future revenues.

3.3.3 Autonomous Agents

Another relevant line of research is the design of autonomous agents. Autonomous agent technology first emerged in the mid to late 1980's in the field of Artificial Intelligence and has since received attention from several researchers in many fields [84]. Particularly of interest is the design of intelligence, in the form of algorithms, of automated bidding and trading agents.

In this realm, there are a number of papers studying bidding algorithms for simultaneous auctions. An interesting feature of these papers is that the algorithms designed and tested target the auction settings in an annual (since 2000) international research competition, namely the Trading Agent Competition (TAC) [79] [73]. TAC invites researchers to design autonomous trading agents and compete under two scenarios "TAC Classic" and "TAC SCM". TAC Classic is a "travel agent" scenario based on complex procurement on multiple simultaneous forward auctions. TAC SCM is a PC manufacturer scenario, involving both forward and reverse auctions, based on sourcing of components, manufacturing of PC's and sales to customers. At a high-level, the design of many successful TAC agents can be summarized as: (i) predict: build a model of the auctions clearing prices, and (ii) optimize: solve for an (approximately) optimal set of bids, given this model. Under the settings of TAC Classic, Greenwald and Boyan [35] formulate the problem of bidding in simultaneous auctions as a stochastic program and solve it using sample average approximation. Candale and Sen [14] compare the performance of a multi-start local search heuristic, based on iteratively updating the bids for each of the items on auction [13], to the top performing algorithms in TAC Classic. Greenwald et al. [6] describe a finalist in the 2003 TAC SCM which uses a similar multi-start local search heuristic as a bidding algorithm. More papers describing the competition and agents can be found on the TAC website [79].

Note also that, some of the research mentioned in the area of transportation procurement auctions are highly relevant to autonomous agent design. The research by Figliozzi [31], Mes et al. [54], and Douma et al. [21] are very good examples.

3.4. A Bid Price Optimization Problem

We study simultaneous transportation procurement auctions from a truckload carrier's perspective. Our goal is neither to design simultaneous auction mechanisms with desirable properties, nor to investigate if the auction mechanisms in place have such properties, nor to characterize equilibrium strategies of carriers in such auctions. Our goal is to develop bidding algorithms for a carrier incorporating anticipated bidding behavior of his competitors. Our approach is similar to the literature discussed in Section 3.3.3 in that the algorithms we develop can be the basis of the intelligence of agent-based transportation management systems.

We take the perspective of a carrier who we assume to have a network which consists of long term contracts. We assume that the carrier wants to complement his network with short term contracts from an auction based spot procurement market in order to recover some of his repositioning costs. In this spot procurement market, the carrier must place bids in multiple independent single lane auctions, i.e. no combinatorial bids, which are run in parallel. These auctions take place at the beginning of every period (e.g. week, month, etc), and involve lanes which must be served in the period(s) that follow . Each of the auctions is a single-round, sealed-bid, first-price auction. We assume that the lanes obtained from the auctions are executed for only a single period, and that the carrier has sufficient capacity to handle all the lanes he gets from the auctions. Another assumption we make is that the cost of traveling between two points is directly proportional to the distance traveled. We also assume that the distances are symmetric and satisfy the triangle inequality.

Without loss of generality, we assume that there is a single auctioneer running all the auctions. The auctioneer may be a single shipper or a third party facilitating the communication between the shippers and the carriers. The following stages take place in sequence:

1. The auctioneer announces which lanes are going to be auctioned.
2. Carriers place bids on the lanes being auctioned.
3. The auctioneer receives the bids and determines which carrier will be given the right to serve each of the lanes.
4. The carrier routes his vehicles to serve the lanes he won

The carrier makes decisions in Stages 2 and 4. The decisions in Stage 2 affects the carrier's revenue while the decisions in Stage 4 affects the carrier's costs. The stochastic optimization model is designed to determine optimal bids in Stage 2 taking into account potential outcomes and thus costs in Stage 4. In order to account for the uncertainty regarding the outcomes of the auctions, the lowest bid of the competing carriers for each lane is modeled as an independent continuous random variable which is uniformly distributed on an interval. We make this simplifying assumption because it is easier for a carrier to determine the range of values of his competitors' bids than to characterize the actual distribution of such bids. In addition, as will be seen later, it is a trivial task to incorporate other probability distributions into our solution approach (see proof of Proposition 3.3).

Let L_0 denote the set of lanes representing the carrier's network. Let $L = \{1, 2, \ldots, m\}$ denote the set of lanes being auctioned. For each lane i, the random variable representing the lowest bid of the competing carriers on lane i is denoted by X_i which is uniformly distributed on the interval $[\ell_i, u_i]$.

The expected profit of the carrier is a function of his bids on the lanes being auctioned. Note that since the carrier knows the lowest (ℓ_i) and highest (u_i) possible values of X_i, it is not necessary for the carrier to bid outside the interval $[\ell_i, u_i]$ for each of the lanes. Since bidding ℓ_i on lane i guarantees winning the lane, the carrier need not bid lower than ℓ_i.

Similarly, since bidding u_i on lane i guarantees losing the lane, the carrier does not need to consider bidding higher than u_i. So, we restrict the carrier's bids to the values within the interval $[\ell_i, u_i]$. Therefore, the optimization problem has the following form:

$$\begin{aligned} \max \quad & \pi(b) = \sum_{S \subseteq L} \Big\{ P(S,b)\ Q(L-S,b)\ \big[R(S,b) - C(S)\big] \Big\} \\ \text{subject to} \quad & b_i \in [\ell_i, u_i] \quad \forall i \in L \end{aligned}$$

where

$$\begin{aligned} b &= \text{Vector of bids for the lanes (decision variables} \\ P(S,b) &= \text{Probability of winning the set of lanes } S \text{ with bids } b \\ Q(L-S,b) &= \text{Probability of losing the set of lanes } L-S \text{ with bids } b \\ R(S,b) &= \text{Revenue obtained from the set of lanes } S \text{ with bids } b \\ C(S) &= \text{Incremental cost of serving the set of lanes } S \end{aligned}$$

Note in the above formulation that the product $P(S,b)Q(L-S,b)$ gives the probability of winning exactly the set of lanes S with a given vector of bids b for the carrier. Since the carrier wins a lane if his bid is lower than the competitor's bid and the competitor's bids are uniformly and independently distributed, we have

$$P(S,b) = \prod_{i \in S} \mathbb{P}\{X_i \geqslant b_i\} = \prod_{i \in S} \left(\frac{u_i - b_i}{u_i - \ell_i} \right). \tag{3.1}$$

Similarly, since the carrier loses a lane if his bid is higher than the competitor's bid, we have:

$$Q(L-S,b) = \prod_{i \in L-S} \mathbb{P}\{X_i \leqslant b_i\} = \prod_{i \in L-S} \left(\frac{b_i - \ell_i}{u_i - \ell_i} \right). \tag{3.2}$$

Since the auctions are assumed to be first price, the revenue that can be obtained from a given set of lanes S with a given vector of bids b is given by:

$$R(S,b) = \sum_{i \in S} b_i. \tag{3.3}$$

Note that the incremental cost of serving the set of lanes in S, denoted by $C(S)$, is independent of the vector of bids b. It however depends on the way the carrier routes his trucks over his existing network and the set of lanes S. In other words, the carrier must solve a routing problem to determine his costs. It is assumed that the carrier solves the aforementioned Lane Covering Problem (LCP) which can be solved in polynomial time (see Section 2.2 of this thesis). We make this assumption in order to avoid having to solve an $\mathcal{NP}$-Complete routing problem and to concentrate our efforts on solving the pricing problem. Also, based on our research on shipper collaboration problem, LCP provides a good approximation of a truckload carrier's costs.

3.5. A Bid Price Optimization Algorithm

The objective function in the formulation given in the previous section involves the enumeration of all possible outcomes of the auctions, i.e. $\forall S \subseteq L$. The number of potential outcomes grows exponentially as number of lanes increases, which makes the problem difficult to solve. Even evaluating the objective function may become computationally prohibitive for large number of lanes let alone solving the bid price optimization problem. Therefore, we focus on designing effective and efficient heuristic algorithms.

The proposed algorithm is based on coordinate search and can be described as follows. Given a vector of bids (with some arbitrary feasible values), the algorithm cycles through the coordinates of the vector. Each component (or coordinate) of the vector is updated, keeping the other components fixed, in a way that results in the maximum improvement in the objective function. This process is repeated until none of the components of the bid vector can be changed by more than a pre-specified value. A more formal description is presented in Algorithm 3.1.

Note that the coordinate search algorithm involves solving the optimization problem $\max\left\{\pi(b) : b_i \in [\ell_i, u_i], b_j = b_j^k \;\; \forall j \in L - \{i\}\right\}$ for a given feasible bid vector b^k and a coordinate i. If this componentwise optimization problem always has a unique solution, then the coordinate search algorithm increases the objective function value each time it moves from one solution to another. Thus, the algorithm never visits the same solution more than

Algorithm 3.1 Coordinate search algorithm.

```
**Given:** Lane set $L = \{1, 2, .., n\}$, intervals $[\ell_i, u_i] \;\; \forall i \in L$, an initial feasible solution vector $b^0$ (e.g. $b_i^0 = u_i \;\; \forall i \in L$), and $\epsilon > 0$.

$k \leftarrow 0$
repeat
  $\gamma \leftarrow false$
  for all $i \in L$ do
    $\overline{b} \leftarrow \arg\max \left\{ \pi(b) : b_i \in [\ell_i, u_i], b_j = b_j^k \;\; \forall j \in L - \{i\} \right\}$
    if $|\overline{b_i} - b_i^k| > \epsilon$ then
      $\gamma \leftarrow true$
      $k \leftarrow k + 1$
      $b_i^k \leftarrow \overline{b_i}$
      for all $j \in L - \{i\}$ do
        $b_j^k \leftarrow b_j^{k-1}$
      end for
    end if
  end for
until $\gamma = false$
```

once, i.e. it does not cycle. Note also that the coordinate search algorithm is a hill-climbing algorithm, which only utilizes local information about the objective function. Therefore, it does not guarantee finding a global optimum solution when the objective function is not concave.

In Section 3.6, we analyze the bid price optimization problem to identify properties of its objective function and its optimal solutions. We first analyze the optimization problem to investigate concavity and other properties of the objective function. Our analysis shows that the objective function is not concave in general but it is concave under certain conditions. Since coordinate search is applicable only if we can maximize the objective function with respect to a single component of the bid vector, we also derive the expression for maximizing the objective function with respect to an arbitrary component of the bid vector. We then use this expression to derive an upper bound on the optimal solutions which may be used in fine tuning of the solution algorithms.

3.6. Analysis

First, we analyze the problem when there are only a small number of lanes on auction since this is expected to be easier than analyzing the problem for an arbitrarily large number of lanes. We then use the obtained results to identify the properties of the problem in general.

3.6.1 Single Lane

Suppose the carrier is bidding on a single lane only, i.e. $L = \{1\}$. There are two possible outcomes. The carrier either wins the lane or loses it. If the carrier loses the auction, he incurs zero additional cost and earns zero additional revenue. If he wins the auction, his additional revenue is equal to his bid b_1.

In this case, the optimization problem to be solved is:

$$\begin{aligned} \max \quad & \left(\frac{u_1 - b_1}{u_1 - \ell_1}\right)(b_1 - C(\{1\})) \\ \text{subject to} \quad & \ell_1 \le b_1 \le u_1 \end{aligned}$$

Note that π is concave in this case. In fact, it is a second degree polynomial with a negative second derivative and roots u_1 and $C(\{1\})$.

$$\begin{aligned} \pi(b) &= \left(\frac{u_1 - b_1}{u_1 - \ell_1}\right)(b_1 - C(\{1\})) \\ &= -\frac{1}{u_1 - \ell_1}(b_1 - u_1)(b_1 - C(\{1\})) \end{aligned}$$

The optimal solution can easily be computed in this case:

$$b_1^* = \begin{cases} \frac{u_1 + C(\{1\})}{2} & \frac{u_1 + C(\{1\})}{2} \in [\ell_1, u_1] \\ \ell_1 & \frac{u_1 + C(\{1\})}{2} < \ell_1 \\ u_1 & \frac{u_1 + C(\{1\})}{2} > u_1 \end{cases} \tag{3.4}$$

See figures 3.1(a), 3.1(b) and 3.1(c) for illustrations.

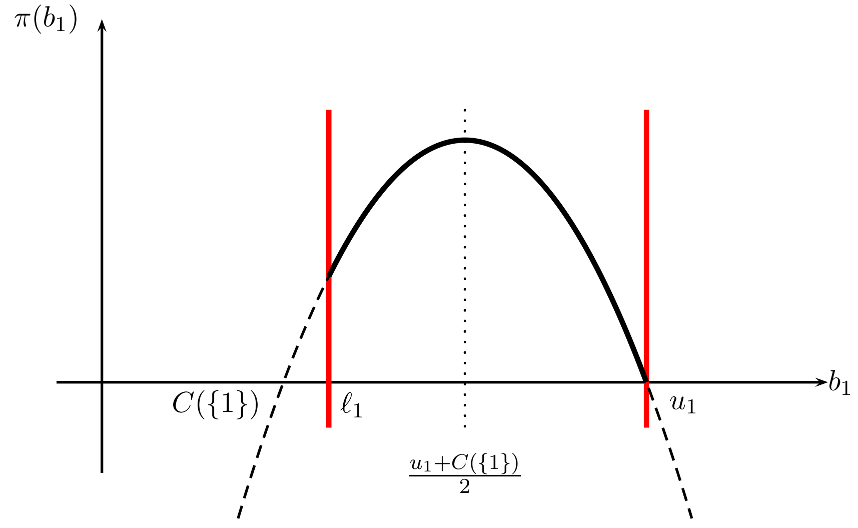
3.6.2 Two Lanes

Here, it is necessary to assume that

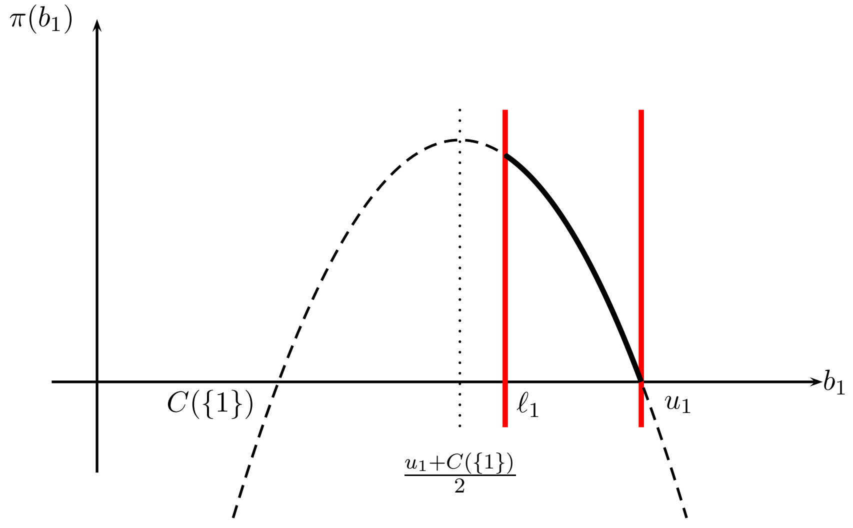
$$C(\{1,2\}) < C(\{1\}) + C(\{2\}),$$

since otherwise the auctions for the two lanes can be analyzed separately as two independent single lane auctions. Let $\Delta = C(\{1\}) + C(\{2\}) - C(\{1,2\})$. In this case, the objective function is a quadratic function which can be written in the form:

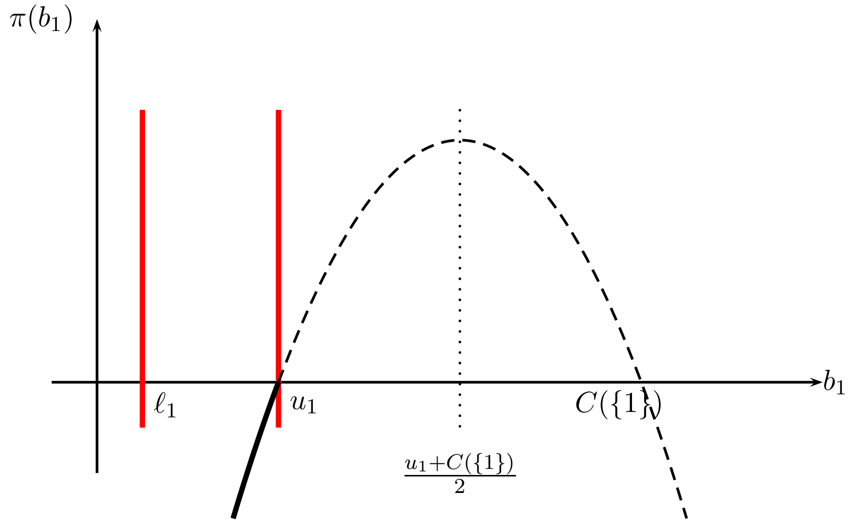
$$\pi(b) = \frac{1}{2} b^T H b + b^T f + g \tag{3.5}$$



(a) The peak falls in the middle of the interval.



(b) The peak falls to the left of the interval.



(c) The peak falls to the right of the interval.

Figure 3.1: Optimal solutions in the single lane case.

where

$$H = \frac{1}{(u_1 - \ell_1)(u_2 - \ell_2)} \begin{bmatrix} -2(u_2 - \ell_2) & \Delta \\ \Delta & -2(u_1 - \ell_1) \end{bmatrix}$$

which is, in fact, the Hessian of $\pi(b)$, and

$$f = \frac{1}{(u_1 - \ell_1)(u_2 - \ell_2)} \begin{bmatrix} (u_2 - \ell_2)C(\{1\}) + u_2(u_1 - \Delta) \\ (u_1 - \ell_1)C(\{2\}) + u_1(u_2 - \Delta) \end{bmatrix}$$

$$g = \frac{u_1\ell_2 C(\{1\}) + \ell_1 u_2 C(\{2\}) - u_1 u_2 C(\{1,2\})}{(u_1 - \ell_1)(u_2 - \ell_2)}$$

Theorem 3.1 *Let $L = \{1, 2\}$ and $\Delta = C(\{1\}) + C(\{2\}) - C(\{1, 2\})$. Then, $\pi(b)$ given by* (3.5) *is concave if and only if*

$$4(u_1 - \ell_1)(u_2 - \ell_2) - \Delta^2 \geqslant 0 \tag{3.6}$$

Proof: It is well known that a real valued function is concave if and only if its Hessian is negative semi-definite. Thus, the function $\pi(b)$ is concave if and only if H is a negative semi-definite matrix. Note that H is a symmetric real-valued square matrix. We are going to make use of the following[1]:

Lemma 3.2 *Let $\mathbf{A}$ be a symmetric real $n \times n$ matrix, and let*

$$\chi(\lambda) = \det(\lambda\mathbf{I} - \mathbf{A}) = \lambda^n + a_{n-1}\lambda^{n-1} + a_{n-2}\lambda^{n-2} + \cdots + a_1\lambda + a_0.$$

Then

1. $\mathbf{A}$ *is positive definite* $\iff$ *all the a_i satisfy $a_i(-1)^{n-i} > 0$.*

2. $\mathbf{A}$ *is positive semidefinite* $\iff$ *all the a_i satisfy $a_i(-1)^{n-i} \geq 0$.*

3. $\mathbf{A}$ *is negative definite* $\iff$ *all the $a_i > 0$.*

4. $\mathbf{A}$ *is negative semidefinite* $\iff$ *all the $a_i \geq 0$.*

5. $\mathbf{A}$ *is indefinite* $\iff$ *none of the previous four conditions on the a_i holds.*

[1] Proof can be found in Sydsaeter and Hammond [72], and Loftin [50].

For H,

$$\begin{aligned}
\chi(\lambda) &= det(\lambda\mathbf{I} - H) \\
&= det\left(\begin{bmatrix} \lambda + \frac{2}{u_1-\ell_1} & \frac{\Delta}{(u_1-\ell_1)(u_2-\ell_2)} \\ \frac{\Delta}{(u_1-\ell_1)(u_2-\ell_2)} & \lambda + \frac{2}{u_2-\ell_2} \end{bmatrix}\right) \\
&= \lambda^2 + 2\left(\frac{1}{u_1-\ell_1} + \frac{1}{u_2-\ell_2}\right)\lambda + \frac{4}{(u_1-\ell_1)(u_2-\ell_2)} - \left(\frac{\Delta}{(u_1-\ell_1)(u_2-\ell_2)}\right)^2 \\
&= \lambda^2 + \underbrace{2\left(\frac{1}{u_1-\ell_1} + \frac{1}{u_2-\ell_2}\right)}_{\geq 0}\lambda + \underbrace{\frac{4(u_1-\ell_1)(u_2-\ell_2) - \Delta^2}{(u_1-\ell_1)^2(u_2-\ell_2)^2}}_{?}
\end{aligned}$$

It follows that H is negative semidefinite $\Leftrightarrow 4(u_1 - \ell_1)(u_2 - \ell_2) - \Delta^2 \geq 0$.

□

If (3.6) does not hold, then H is indefinite which means that $\pi(b)$ is saddle shaped. If $\pi(b)$ is concave, the optimal solution may be in the interior of the feasible region. If $\pi(b)$ is saddle shaped, then the optimal solution is certainly on the boundary of the feasible region. Also, when $\pi(b)$ is concave, the coordinate search algorithm is guaranteed to find the optimal solution. Otherwise, coordinate search may get stuck in a local optimum failing to find the global optimum.

3.6.3 Componentwise Optimization with Arbitrary Number of Lanes

In order for the coordinate search to work, we need to be able to change a given component of the bid vector, keeping all other component fixed, such that the increase in the objective function is maximized. In other words, for each i^{th}component of b we treat $\pi(b)$ as if it is only a function of b_i and maximize it with respect to b_i. We first write $\pi(b)$ as a function of a single bid to see if we can easily maximize it with respect to that bid.

Proposition 3.3 *If the objective function $\pi(b)$ is written as a function of a single bid b_i, we get:*

$$\begin{aligned}
\pi_i(b) = \left(\frac{u_i - b_i}{u_i - \ell_i}\right)&\left(b_i - \sum_{\substack{V\subseteq L \\ i\notin V}} P(V,b)\ Q(L - V - \{i\}, b)\,(C(V + \{i\}) - C(V))\right) \\
&+ \sum_{\substack{V\subseteq L \\ i\notin V}} P(V,b)\ Q(L - V - \{i\}, b)[R(V,b) - C(V)]
\end{aligned} \tag{3.7}$$

Proof:

Recall that

$$\begin{aligned} P(S,b) &= \prod_{i\in S} \mathbb{P}\{X_i \geqslant b_i\} = \prod_{i\in S}\left(\frac{u_i-b_i}{u_i-\ell_i}\right) \\ Q(L-S,b) &= \prod_{i\in L-S} \mathbb{P}\{X_i \leqslant b_i\} = \prod_{i\in L-S}\left(\frac{b_i-\ell_i}{u_i-\ell_i}\right) \\ R(S,b) &= \sum_{i\in S} b_i \end{aligned}$$

Then,

$$\begin{aligned} \pi(b) &= \sum_{S\subseteq L} P(S,b)\; Q(L-S,b)\; [R(S,b)-C(S)] \\ &= \sum_{\substack{U\subseteq L\\ i\in U}} P(U,b)\; Q(L-U,b)\; [R(U,b)-C(U)] \\ &\quad + \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V,b)\; [R(V,b)-C(V)] \\ &= P(\{i\},b) \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [R(V+\{i\},b)-C(V+\{i\})] \\ &\quad + Q(\{i\},b) \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [R(V,b)-C(V)] \\ &= P(\{i\},b) \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [R(V+\{i\},b)-C(V+\{i\})] \\ &\quad + [1-P(\{i\},b)] \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [R(V,b)-C(V)] \\ &= P(\{i\},b) \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [b_i+R(V,b)-C(V+\{i\})+C(V)-C(V)] \\ &\quad + [1-P(\{i\},b)] \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [R(V,b)-C(V)] \\ &= P(\{i\},b) \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [b_i-C(V+\{i\})+C(V)] \\ &\quad + P(\{i\},b) \sum_{\substack{V\subseteq L\\ i\notin V}} P(V,b)\; Q(L-V-\{i\},b)\; [R(V,b)-C(V)] \end{aligned}$$

$$+ [1 - P(\{i\}, b)] \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)\ [R(V, b) - C(V)]$$

$$= P(\{i\}, b) \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)\ [b_i - C(V + \{i\}) + C(V)]$$

$$+ \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)\ [R(V, b) - C(V)]$$

$$= P(\{i\}, b) \left(\underbrace{b_i \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)}_{=1} \right.$$

$$\left. - \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)\ [C(V + \{i\}) - C(V)] \right)$$

$$+ \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)\ [R(V, b) - C(V)]$$

$$= P(\{i\}, b) \left(b_i - \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b) \left(C(V + \{i\}) - C(V) \right) \right)$$

$$+ \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)[R(V, b) - C(V)]$$

Since in our case $P(\{i\}, b) = \frac{u_i - b_i}{u_i - \ell_i}$ we get:

$$\pi_i(b) = \left(\frac{u_i - b_i}{u_i - \ell_i} \right) \left(b_i - \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b) \left(C(V + \{i\}) - C(V) \right) \right)$$

$$+ \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)[R(V, b) - C(V)]$$

□

Let

$$\sigma_i(b) = \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b) \left(C(V + \{i\}) - C(V) \right) \tag{3.8}$$

Note that $\pi_i(b)$ is equal to the sum of a constant and a second degree concave polynomial with roots u_i and $\sigma_i(b)$. Then, given an arbitrary feasible vector of bids b and an index i, the best improvement in the objective function is achieved, keeping all the other bids constant, when we set b_i to:

$$\overline{b_i} = \begin{cases} \frac{u_1+\sigma_i(b)}{2} & \frac{u_1+\sigma_i(b)}{2} \in [\ell_i, u_i] \\ \ell_i & \frac{u_1+\sigma_i(b)}{2} < \ell_i \\ u_i & \frac{u_1+\sigma_i(b)}{2} > u_i \end{cases} \tag{3.9}$$

It is important to note that $\overline{b_i}$ is the unique maximizer of $\pi_i(b)$ on $[\ell_i, u_i]$. Because of the uniqueness of the componentwise maximizer, the coordinate search algorithm moves from one solution to another if and only if the objective function value increases as a result of that move. Therefore, the algorithm never cycles.

Note that $\sigma_i(b)$ is a measure of synergy between lane i and the rest of the lanes. It is equal to the expected increase in the cost of the carrier if the carrier wins lane i given his bids on the other lanes. It can also be used to find an upper bound on the value of the optimal bids. The worst scenario when lane i is added to a given set V of lanes is that it cannot be put in a cycle with any of the other lanes and thus has to be served by itself, i.e. in a cycle which consists of a loaded move along the lane and an empty move in the opposite direction. Let c_i be the cost of traveling one-way along the lane. So, the worst case incremental cost of serving the lane if the carrier wins the auction is equal to $2c_i$. Mathematically, this means that:

$$C(V + \{i\}) - C(V) \leqslant 2c_i$$

Then,

$$\sigma_i(b) \leqslant \sum_{\substack{V \subseteq L \\ i \notin V}} P(V, b)\ Q(L - V - \{i\}, b)\ 2c_i = 2c_i$$

and

$$\overline{b_i} = \frac{u_i + \sigma_i(b)}{2} \leqslant \frac{u_i + 2c_i}{2}$$

Therefore, an upper bound for the optimal bid is identified. Note that the cost of serving a single lane when the carrier does not have an existing network is also equal to $2c_i$. This information can be useful for speeding up the algorithm in the following way. For each lane, the initial bid value can be obtained by replacing $C(\{i\})$ with $2c_i$ in (3.4). Then, the coordinate search can be performed ignoring the coordinates for which the initial values are equal to their lower bounds.

3.7. Algorithm Implementation

Coordinate search starts with a feasible solution and iteratively obtains better solutions by setting the bid for each lane i, denoted b_i, to $\overline{b_i}$ given by (3.9) which requires the computation of $\sigma_i(b)$ as in (3.8). The algorithm makes use of the cost of serving each and every subset of L over and over again. Therefore, given a feasible solution $\hat{b}$ and lane i, in order to compute $\sigma_i(\hat{b})$ efficiently, we need a method to obtain quickly the service cost of each subset of L. We also need to be able to identify the subsets of L to which lane i belongs and the subsets of L to which lane i does not belong. Such a method is also needed for evaluating the objective function of the bid price optimization problem.

The remedy we propose is solving LCP on every subset of L and storing the values we obtain in a one-dimensional array. However, we cannot store the values in the array in an arbitrary order. We need to map each subset of L to a location in the array, so that we can access the solution of LCP on a given subset of L in an efficient manner. We can achieve this by using a hash function defined as follows.

- To each lane $i \in \{1, 2, ..., n\}$, assign a hash value of 2^{i-1}.
- Then, the hash value of a subset S of L is the sum of the hash values of lanes in that subset. The location of $C(S)$ in the array is equal to the hash value of S.

Note in this scheme that no two subsets of L have the same hash value and all the hash values between 1 and 2^n correspond to a subset of L. In order words, we have a 1-to-1 mapping of subsets of L to locations in the array. The hash value of a subset provides an intuitive and extremely efficient way of determining which lanes belong to that subset. Consider the binary equivalent of the hash value of an arbitrary subset S of L. Then, $i \in S$ if the i^{th} digit of the binary equivalent is equal to 1 and $i \notin S$ if the i^{th} digit of the binary equivalent is equal to 0. This structure facilitates the computation of $P(S,b)$,$Q(L-S,b)$, and $R(S,b)$ given by (3.1),(3.2), and (3.3), respectively.

In addition, the ordering of the subsets in the array follows a pattern which makes it easy, for a given lane, to identify the subsets which include that lane and the subsets which do not include that lane. For instance, Lane 1 is included in subsets with hash values 1, 3, 5, 7,... and not included in subsets with hash values 2, 4, 6, 8,... . Lane 2 is included in subsets with hash values 2, 3, 6, 7, 10, 11,... and not included in subsets with hash values 1, 4, 5, 8, 9, 12, 13,... . In general,

- $i \notin S$ if $hash(S) \in \{s \in \mathbf{Z} \mid 2k \cdot 2^{i-1} \leqslant s < (2k+1) \cdot 2^{i-1}$ for $k = 0,1,2,3,..,2^{|L|-i-1}\}$
- $i \in S$ if $hash(S) \in \{s \in \mathbf{Z} \mid (2k+1) \cdot 2^{i-1} \leqslant s < (2k+2) \cdot 2^{i-1}$ for $k = 0,1,2,3,..,2^{|L|-i-1}\}$

The hash function scheme we discussed above enables us to design an efficient implementation of the coordinate search algorithm. In the next section, we discuss the performance of our algorithm and its impact on carrier profits.

3.8. Computational Experiments

3.8.1 Performance of Coordinate Search Algorithm

We have shown that the objective function $\pi(b)$ of our optimization problem is not concave in general and therefore may have multiple local optima. Our coordinate search based optimization algorithm may get stuck at a local optimum failing to find the global optimal solution of our optimization problem.

In order to measure the effectiveness of our algorithm, we set up computational experiments to compare the quality of its solutions with the solutions obtained by a well known global optimization solver BARON [74, 67]. BARON stands for "Branch And Reduce Optimization Navigator". It "combines constraint propagation, interval analysis, and duality in its *reduce* arsenal with enhanced *branch and bound* concepts as it winds its way through the hills and valleys of complex optimization problems in search of global solutions [5]." We ran BARON through the NEOS server for optimization [60] where it is publicly available.

We used 50 randomly generated Euclidean problem instances inside a 300×300 region for the experiment. In each of these instances, the number of lanes being auctioned was equal to 10 and the carrier did not have an existing network. The bounds of the interval used in the optimization problem were randomly determined such that $[\ell_i, u_i] \subseteq [0, 4c_i]$ for each lane i. BARON was set to run with default parameter settings, e.g. it was set to terminate after 1000 CPU seconds, or when the optimality gap was smaller than 10%. Our algorithm started with $b_i = u_i$ for all i and was set to terminate when it no longer could move more than a distance of 1×10^{-6} from the current solution.

In all of the instances, BARON terminated after 1000 CPU seconds and returned the solution it computed during its preprocessing phase as the best solution found. Our algorithm returned the same solutions as BARON in all of the instances with CPU times of under 5 seconds, including instance generation and solving LCPs for all possible subsets and writing input files to be used by BARON, where the coordinate search portion takes negligible CPU time. It is important to note that the final optimality gaps in BARON's branch and bound trees were huge. These huge gaps despite intensive computational efforts demonstrate the inherent difficulty in computing verifiably global optimal solutions to our problem. On the other hand, the fact that our algorithm returned the same solution as BARON is an indication that our algorithm is an efficient way of computing good if not optimal solutions to the problem.

3.8.2 Simulation Study

In order to examine the impact of using our optimization based simultaneous bidding algorithm, we simulated a transportation procurement marketplace over 52 periods under different settings. This simulated marketplace has the following properties.

1. Origin-destination pairs come from 270 fixed points which are randomly determined in a 1.0×1.0 square region at the beginning of the simulation.

2. Each point has two coordinates and the distance between any two points is computed using the Euclidean metric.

3. The entire square region is divided into 9 (3 by 3) equal sized squares. The regions are referred to as South West (SW), South (S), South East (SE), West (W), Central (C), East (E), North West (NW), North (N), North East (NE).

4. There is a single auctioneer who runs 10 simultaneous independent single-lane auctions in each period.

5. Each of these auctions:

   (a) is for a contract of length equal to one period.

   (b) has no reserve price.

6. There are two carriers competing for the contracts.

   (a) One of the carriers uses our algorithm to determine his bids and will be referred to as SB (smart bidder) from this point on. For each lane i, SB assumes that the other carrier's bid is uniformly distributed on the interval $[\ell_i, u_i]$. The bounds of the interval are such that $\frac{\ell_i}{c_i} \in \{0.5, 1.0\}$ and $\frac{u_i}{c_i} \in \{1.5, 2.0, 2.5\}$ where c_i is the one-way cost of traversing lane i.

   (b) The other carrier computes the marginal, i.e. insertion, cost of a lane and always adds a 40% markup to the marginal cost to determine his bid on that lane. We will refer to this carrier as MM.

   (c) Both carriers have existing networks consisting of long term contracts. The size of a carrier's network is equal to either 30, 45, or 90 lanes.

   (d) The carriers' networks may be concentrated in a certain region. In order to control this, the networks are generated based on given probabilities of having an origin or destination in each region.

   (e) For both carriers, the revenue from each lane in the existing network is set to the cost of the network divided by the network size so that the net profit at the beginning of the simulation is equal to zero. Therefore, the profits reported in the tables below come from the lanes won in the auctions.

(f) Both carriers place bids on every lane being auctioned in the marketplace.

(g) Both carriers compute operating costs using LCP.

(h) Both carriers refrain from combining lanes obtained in different periods, i.e. LCP is solved on the carriers network and the lanes coming from a single period's auctions.

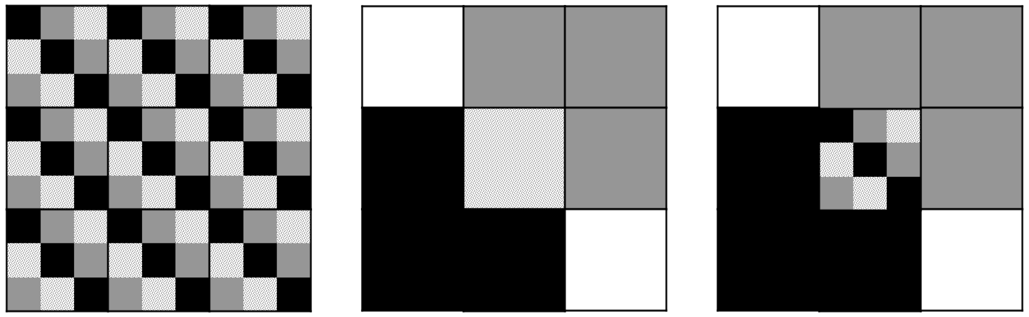
The markup percentage we use for determining MM's bids, and the network sizes of the carriers are based on our experiences in practical settings. Even though we assumed that there is only a single competitor of SB in the market, MM sufficiently represents several competitors from SB's perspective. In the simulation setting, the competitor MM is not completely naive. The assumption that MM uses LCP to compute operating costs attributes a degree of sophistication to MM. In practice, it is not unusual to encounter carriers with less sophisticated vehicle routing technologies.

Three sets of experiments, which combine the settings mentioned above, have been conducted. These are:

E1. **Similar carrier networks:** Each carrier may have an origin or destination with equal probability in each of the regions. So may the auction contracts.

E2. **Disjoint regional carrier networks:** SB's origin-destination pairs come only from regions SW, S and W (with equal probability). MM's origin-destination pairs come only from regions E, N, and NE (with equal probability). For the lanes being auctioned, the probability of an origin or destination is (arbitrarily) set to 68% for region C and 4% for each of the rest of the regions. Note that the carrier networks do not overlap.

E3. **Partially overlapping regional carrier networks:** SB's origin-destination pairs come only from regions SW, S, W, C (with equal probability). MM's origin-destination pairs come only from regions C, E, N, and NE. For the lanes being auctioned, the probability of an origin or destination is 68% for region C and 4% for each of the rest of the regions. Note that the carrier networks overlap in region C.


(a) E1: Similar carrier networks.
(b) E2: Disjoint regional carrier networks.
(c) E3: Partially overlapping regional carrier networks.

Legend: ⬛ SB Network ▩ MM Network ░ Auctions

Figure 3.2: Concentrations of carrier networks and auctioned lanes for the simulation study.

Figure 3.2 illustrates the concentrations of carrier networks and auctioned lanes in the experiments. The summarized results of the experiments are listed in Tables 3.1, 3.2, and 3.3. The statistics reported are average carrier profits, average carrier profit margins, and average number of auctions won by the carriers. The profit amounts reported are the sums of the differences between the revenues and the costs of the carriers over all periods. Note that since we set the carriers' revenues from their existing networks equal to the costs of their networks, they incur zero profit without the lanes from the auctions. So, the profit amounts reported come solely from the lanes being auctioned. The profit margin of a carrier is computed by dividing the total profit by the total cost, both accumulated during the course of the simulation. Because of the way we initialize carrier profits and the way we compute the profit margin, the profit margin of a carrier decreases as his networks size increases. This is natural since a carrier with a larger network needs more lanes to recover his repositioning costs than a carrier with a smaller network.

Each of the Tables 3.1, 3.2, and 3.3 consists of two parts. In the first part, the averages are taken over different carrier network size combinations. In the second part, the averages are taken over different uniform distribution intervals. In what follows, we analyze and interpret these tables.

3.8.2.1 Similar Carrier Networks (E1)

In this experiment, both carriers' networks are evenly distributed over all of the regions. That is, the carrier networks are randomly generated using equal probability of having an origin or destination in each of the regions. Also, the origins and destinations of the lanes being auctioned are equally likely to be in any of the regions.

The results are summarized in Table 3.1, which is composed of two parts. Table (a) contains average values of carrier profits, carrier profit margins, and the number of auctions won by each carrier computed over different distribution intervals. We observe that SB almost always outperforms MM in terms of profit margin. SB also achieves higher profits and wins more auctions when his network is not smaller than MM's network. SB has an advantage over MM because our algorithm allows him to effectively exploit potential synergies between his network and the lanes being auctioned. However, having larger network enables MM to overcome the advantage of SB over MM. In this case, we conclude that economies of scale is more important than economies of scope.

Table 3.1(b) demonstrates the effect of the bounds of the interval used by SB on the carriers' profits. Recall that for each lane $i \in L$, the lower bound ℓ_i and the upper bound u_i of the interval is determined such that LBfactor= $\frac{\ell_i}{c_i} \in \{0.5, 1.0\}$ and UBfactor= $\frac{u_i}{c_i} \in \{1.5, 2.0, 2.5\}$, respectively. The first thing to note here is that the difference between LBfactor between 0.5 and 1.0 is negligible. On the other hand, we observe that the use of a smaller UBfactor enables SB to win more auctions than MM, but leads to a decrease in the profits and profit margins of both carriers. This is an interesting and counter-intuitive observation. Since SB and MM are competing carriers in the same market, one would expect to see an increase in MM's profits if SB's decisions cause his profits to decrease, and vice versa. However, we see that the carriers' strategies impact the whole market. By using a smaller UBfactor, SM is bidding more aggressively and increasing his chances of winning auctions while forgoing some of the profits. This situation is analogous to a price cut initiated by SB to increase his market share. In the end, SB increases his market share but ends up with less profit because of disproportionate increase in his costs. Meanwhile, the shippers benefit from increased competition because they spend less on transportation

Table 3.1: E1: Similar carrier networks.

(a) Network size.

Network Size		**Profit**			**Margin**			**#Auctions won**		
SB	**MM**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**
30	30	$46.53	$32.17	$14.36	3.83%	2.65%	1.19%	296.2	223.8	72.3
30	45	$48.81	$51.00	-$2.19	4.09%	2.94%	1.15%	274.0	246.0	28.0
30	90	$50.66	$53.15	-$2.50	4.29%	1.69%	2.61%	259.7	260.3	- 0.7
45	30	$59.55	$37.52	$22.03	3.47%	3.17%	0.30%	308.5	211.5	97.0
45	45	$47.30	$39.39	$7.91	2.81%	2.36%	0.45%	247.0	273.0	- 26.0
45	90	$41.60	$50.10	-$8.50	2.49%	1.56%	0.93%	232.7	287.3	- 54.7
90	30	$68.70	$32.03	$36.67	2.25%	2.49%	-0.24%	306.3	213.7	92.7
90	45	$65.37	$30.61	$34.76	2.15%	1.67%	0.48%	285.5	234.5	51.0
90	90	$63.26	$34.17	$29.09	2.08%	1.06%	1.02%	273.0	247.0	26.0

(b) Uniform distribution interval.

Interval		**Profit**			**Margin**			**#Auctions won**		
LBfactor	**UBfactor**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**
0.5	1.5	$34.50	$20.42	$14.08	1.87%	1.14%	0.73%	358.8	161.2	197.6
0.5	2.0	$62.70	$38.28	$24.43	3.54%	2.08%	1.45%	275.6	244.4	031.1
0.5	2.5	$66.18	$62.14	$ 4.03	3.73%	3.35%	0.38%	191.7	328.3	- 136.7
1.0	1.5	$34.66	$19.94	$14.72	1.86%	1.11%	0.75%	360.2	159.8	200.4
1.0	2.0	$63.17	$37.76	$25.41	3.56%	2.06%	1.50%	276.6	243.4	033.1
1.0	2.5	$66.62	$61.55	$ 5.08	3.75%	3.31%	0.44%	192.4	327.6	- 135.1

services.

3.8.2.2 Disjoint Regional Carrier Networks (E2)

In this experiment, the carriers have disjoint regional networks. The origins and destinations of SB's network lanes are all located in regions SW, S, or W. The origins and destinations of MM's network lanes are all located in regions E, N, or NE. In the simulation, the networks of the carriers are randomly generated such that SB's network has an origin or destination in any of the regions SW, S or W with probability $\frac{1}{3}$ and MM's network has an origin or destination in any of the regions E,N, or NE with probability $\frac{1}{3}$. For the lanes being auctioned, the probability of an origin or destination is 68% for region C and 4% for each of the rest of the regions. Hence, most of auctions come from the central region and complement both carriers' networks.

In this case, we expect the carriers to compete more fiercely than in the case of E1 for the lanes being auctioned. Since the lanes being auctioned complement the carriers' networks, we also expect the ability to exploit synergies to be critical. In other words, we expect SB to perform much better than MM under this configuration.

The results presented in Table 3.2 confirm our expectations. There is a significant difference between the performance of SB and MM. SB outperforms MM in terms of all three statistics, i.e. profit, profit margin, and the number of auctions won, regardless of network size, even when SB has a network of 30 lanes and MM has a network of 90 lanes. There is also an increase in SB's profit margins compared to E1 (see Table 3.1).

Note that the bounds of the interval used by SB has similar effects on carriers' statistics. Again, note that varying LBfactor between 0.5 and 1.0 makes an insignificant difference. Similarly, observe also that the use of a smaller UBfactor enables SM to win more auctions than MM, but leads to a decrease in the profits and profit margins of both carriers. However, UBfactor has a greater impact on SB's profits. Due to the characteristics of carrier networks and the lanes being auctioned, the carriers must carefully choose which lanes they want and which lanes they do not want. If SB bids too aggressively (by using a small UBfactor), he ends up with lanes he should not have won because they have little synergies with his

Table 3.2: E2: Disjoint regional carrier networks.

(a) Effect of carrier network size

Network Size		**Profit**			**Margin**			**#Auctions won**		
SB	**MM**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**
30	30	$47.20	$10.70	$36.49	5.47%	1.58%	3.89%	467.3	52.7	414.7
30	45	$39.25	$16.39	$22.87	4.74%	1.74%	2.99%	419.5	100.5	319.0
30	90	$40.39	$18.13	$22.26	4.87%	0.97%	3.89%	416.2	103.8	312.3
45	30	$45.29	$12.38	$32.91	4.00%	2.01%	1.99%	452.7	67.3	385.3
45	45	$47.39	$11.02	$36.37	4.17%	1.24%	2.93%	459.2	60.8	398.3
45	90	$43.73	$15.75	$27.99	3.92%	0.86%	3.06%	425.8	94.2	331.7
90	30	$47.52	$11.61	$35.91	2.33%	1.56%	0.78%	447.8	72.2	375.7
90	45	$45.35	$13.16	$32.19	2.24%	1.27%	0.97%	445.0	75.0	370.0
90	90	$49.48	$ 9.12	$40.36	2.41%	0.50%	1.91%	472.8	47.2	425.7

(b) Effect of uniform distribution interval

Interval		**Profit**			**Margin**			**#Auctions won**		
LBfactor	**UBfactor**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**
0.5	1.5	$10.12	$4.23	$5.90	0.79%	0.41%	0.38%	485.9	34.1	451.8
0.5	2.0	$51.89	$14.02	$37.87	4.34%	1.41%	2.93%	438.4	81.6	356.9
0.5	2.5	$75.27	$22.86	$52.41	6.43%	2.26%	4.17%	404.7	115.3	289.3
1.0	1.5	$9.33	$4.21	$5.13	0.74%	0.41%	0.33%	486.0	34.0	452.0
1.0	2.0	$49.14	$12.27	$36.87	4.12%	1.22%	2.89%	445.6	74.4	371.1
1.0	2.5	$74.64	$21.26	$53.39	6.36%	2.10%	4.25%	410.3	109.7	300.7

existing network.

3.8.2.3 Partially Overlapping Regional Carrier Networks (E3)

In this experiment, the origins and destinations of SB's network lanes are all located in regions SW, S, W, or C while the origins and destinations of MM's network lanes are all located in regions C, E, N, and NE. In the simulation, the networks of the carriers are randomly generated such that SB's network has an origin or destination in any of the regions SW, S, W or C with probability $\frac{1}{4}$ and MM's network has an origin or destination in any of the regions C, E, N, or NE with probability $\frac{1}{4}$. A lane begin auctioned has an origin or destination in region C with probability 64% and in the other regions with probability 4% each.

Note that this configuration is a crossover between the configurations of the experiments E1 and E2. Therefore, we expect the results to be somewhere between the results of the two previously discussed experiments.

Once again, the results summarized in Table 3.3 confirm our expectations. SB outperforms MM regardless of network size, however the difference between the two carriers is not as great as in E2. The ability to exploit synergies is still significant but not as critical as in E2.

The bounds of the interval used by SB has similar effects on carriers' statistics (see Table 3.3(b)). We again observe that varying LBfactor between 0.5 and 1.0 makes little difference. Meanwhile, the use of a smaller UBfactor enables SB to win more auctions than MM, but leads to a decrease in the profits and profit margins of both carriers.

The overall results of our simulation experiments indicate that our method gives SB an advantage in terms of achieving economies of scope by exploiting synergies between lanes. The performance of SB, who uses our algorithm, is affected by his existing network, the lanes being auctioned, and MM's existing network. In general, the larger the existing network of MM, the better he performs against SB. Thus, MM can outperform SB if his network is sufficiently larger than SB's. The difference between the two networks does not have to be too large if the lanes being auctioned are in the area where MM's existing network

Table 3.3: E3: Partially overlapping regional carrier networks

(a) Effect of network size.

Network Size		**Profit**			**Margin**			**#Auctions Won**		
SB	**MM**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**
30	30	\$30.46	\$23.96	\$6.51	3.86%	2.98%	0.88%	299.8	220.2	79.7
30	45	\$27.37	\$25.10	\$2.27	3.52%	2.15%	1.37%	276.2	243.8	32.3
30	90	\$27.43	\$25.65	\$1.79	3.51%	1.25%	2.26%	277.2	242.8	34.3
45	30	\$32.54	\$21.21	\$11.33	2.98%	2.56%	0.42%	290.7	229.3	61.3
45	45	\$32.01	\$25.14	\$6.87	2.95%	2.20%	0.75%	287.2	232.8	54.3
45	90	\$34.89	\$21.35	\$13.54	3.19%	1.07%	2.11%	305.5	214.5	91.0
90	30	\$42.33	\$11.02	\$31.32	2.12%	1.45%	0.66%	350.5	169.5	181.0
90	45	\$34.56	\$12.44	\$22.11	1.72%	1.18%	0.54%	354.0	166.0	188.0
90	90	\$36.87	\$17.28	\$19.59	1.84%	0.89%	0.95%	365.0	155.0	210.0

(b) Effect of distribution interval

Interval		**Profit**			**Margin**			**#Auctions won**		
LBfactor	**UBfactor**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**	**SB**	**MM**	**diff**
0.5	1.5	\$10.79	\$11.02	-\$0.23	0.85%	0.98%	-0.14%	366.9	153.1	213.8
0.5	2.0	\$39.44	\$20.27	\$19.17	3.34%	1.76%	1.59%	308.8	211.2	97.6
0.5	2.5	\$52.70	\$31.80	\$20.89	4.56%	2.65%	1.91%	249.6	270.4	-20.9
1.0	1.5	\$10.69	\$10.92	-\$0.24	0.84%	0.97%	-0.13%	366.8	153.2	213.6
1.0	2.0	\$38.63	\$19.69	\$18.94	3.27%	1.71%	1.56%	310.8	209.2	101.6
1.0	2.5	\$49.89	\$32.26	\$17.63	4.96%	2.81%	2.15%	241.7	278.3	-36.6

is concentrated. Our results also indicate that the pricing decisions of a carrier may have system-wide effects. A carrier with an overly aggressive bidding strategy risks degrading the profitability of not only himself but also the other carriers in the market. We observed that the upper bound of the uniform distribution interval determines the level of aggressiveness of SB. The lower the upper bound, the more aggressive is SB. Therefore, choosing the right interval is of critical importance for the performance of the method. Note that the interval represents SB's prediction of his competitor MM's behavior. Thus, the effectiveness of the method depends on how well SB can predict MM's behavior. The carrier should make sure sufficient prediction technology is in place to get the most out of the bid price optimization approach we propose.

CHAPTER IV

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we study research problems arising from collaboration and simultaneous auctions in the procurement of truckload transportation services. We study collaboration from the shippers' perspective whereas we study simultaneous auctions from the carriers' perspective. In both cases, our approach is formulating optimization problems and developing efficient and effective solution algorithms for solving these problems. We then demonstrate the positive impact of our methods through case studies and/or computational experiments. In the problems we study, the central issue is identifying and exploiting synergies between the lanes of a carrier's network.

4.1. Shipper Collaboration

In Chapter 2, we study collaboration in the procurement of truckload transportation services from the shipper's perspective. Collaborating shippers extend their focus beyond traditional corporate boundaries and consider the system-wide effects of their operations. The key to the success of collaborative transportation procurement is understanding what drives the carriers costs and working jointly towards eliminating them. In effect, shippers solve a problem of the carrier by collaborating and expect to receive better rates in return. Our research introduces the resulting optimization problem (LCP) and show that it can be solved in polynomial time. We also show that the length, cardinality, and time constrained variants of LCP are $\mathcal{NP}$-Complete. We develop fast solution algorithms which combines cycle generation and greedy construction heuristics with local improvement and returns high quality solutions. A real-world case study shows that our algorithms can be used in reducing truckload transportation costs of real-world companies.

In general, we have assumed that either minimizing the duration for covering lanes is a good determinant on the goodness of a solution or used an explicit carrier pricing model. In the auction models used in truckload transportation services procurement, it is assumed that the carriers will price bundle of lanes externally. The optimization technology we have developed here can be used by the shippers to create these bundles for such a procurement auction, used as a tool when negotiating contracts with carriers or evaluating the value of collaboration.

In our description of LCP and its variants, we require all of the lanes to be covered by cycles of minimum total length. By doing so, we implicitly assume that the non-collaborative solution is just a direct loaded move with an empty backhaul. Practically, the move would be handled by a contract carrier as a one-way move. Consequently, it may not be the best option to cover all the lanes by cycles, especially if some of the lanes have very good one-way rates. Thus, the shippers must carefully decide which lanes should be covered by cycles. The problem of deciding which lanes should go into collaboration can easily be integrated into our formulation of CCLCP, LCLCP, and TCLCP by simply setting the cost of each cycle with a single lane equal to the one-way rate quoted by the contract carrier.

Even though our solution approach addresses many practical considerations, it does not address all of them. An interesting variant of LCP arises if each shipper has a certain limit on the number of shippers he wants to collaborate with. In that case, we need to come up with a different formulation, which is not trivial. It is also not trivial to modify our algorithm in a way that ensures feasibility of the solutions found. In defining LCP and its variants, we ignored the U.S. Department of Transportation's Hours of Service regulations for drivers, which is common practice, however unfortunate, among academics. An interesting and very challenging question is whether we could alter our algorithms to properly account for those regulations, especially when time windows are present.

For a shipper network to survive, given a price for a bundle of lanes (created either by the shippers or the carriers), mechanisms for allocating the cost among the shippers and membership rules managing the collaboration must be established. The existence of desirable cost allocation mechanisms is well studied in the cooperative game theory literature. However, there is no one-size-fits-all recipe to determine the existence of such allocation mechanisms. In addition, we need to go one step beyond the existence question and design mechanisms which can be used by the shippers [29].

4.2. Bidding in Simultaneous Transportation Procurement Auctions

In Chapter 3, we study simultaneous transportation procurement auctions from a carrier's perspective. The shippers use auctions for transportation procurement to discover carrier(s) whose network has the most synergy with his lane(s). A carrier participating in these auctions must make dynamic pricing decisions taking into account his current network, the lanes being auctioned, and his competitors' bids. The main challenges stem from the uncertainty involved with the set of lanes the carrier will be serving after the auctions. In order to account for the uncertainty, we assume a that the competitors' lowest bid on each lane is an independent uniform random variable. Based on this assumption, we formulate a stochastic optimization problem and propose a coordinate search algorithm for solving it. Our computational experiments indicate that the coordinate search algorithm produces good solutions to our problem. We observe that our bidding algorithm helps the carrier outperform his competitors, especially in situations where exploiting synergies is critical. We also observe that a carrier's bidding strategy may have system-wide effects.

In our problem description, we ignored the capacity considerations of the carrier in terms of the number of lanes that he can handle with his existing fleet. If the carrier wins too many auctions, the carrier may fail to serve all of his contracts with his existing fleet and incur additional costs. For instance, the carrier may have to abandon some of his contracts, which may cost the carrier in penalty fees or customer good will. Alternatively, the carrier may have to subcontract some of his lanes. In other words, for any subset of $C \subseteq L$ which the carriers fails to serve with his existing fleet, the cost of serving S, $C(S)$, is no longer equal to the solution of LCP on that subset. However, we can easily incorporate additional costs incurred by the carrier as a result of exceeding capacity in our formulation. Let $\mathfrak{S}$ be the set of subsets of L which cannot be served entirely using only the existing fleet of the carrier. Then, after solving LCP on every subset S of L and determining fleet capacity requirements, we add the relevant costs on top of the LCP cost for all $S \in \mathfrak{S}$ to get the right values of $C(S)$.

The carrier may want to limit the possibility of his fleet capacity falling short in fulfilling his contracts, because of excessive penalty fees or subcontracting costs. More importantly, the carrier may not be allowed to subcontract or subcontracting may ruin the carrier's reputation. In order address capacity considerations, in addition to the cost modifications we discussed above, we can add a chance constraint to our optimization problem. Based on the assumption that the lowest bids on the auctions are independently distributed, this chance constraint has the following form:

$$\sum_{S \in \mathfrak{S}} P(S, b) \cdot Q(L - S, b) \leq \alpha$$

where $\alpha \in [0, 1]$ is a parameter determined by the carrier. The value of α is expected to be close to 0, e.g. 0.2, 0.1, 0.05, 0.02, etc. Since the value of α depends on difficult to quantify factors such as carrier preferences and loss of customer good will, determining the right value of α is more of an art than science and is not trivial.

Our optimization problem is built on the assumption that the probability distribution of the lowest bid placed by the competitors on each lane is an independent uniform random variable. In other words, we are given the lower and upper bounds of competitors' bids and any value in between is equally likely. In practice, the carrier may not have that information, so we may need a way to compute the bounds. Therefore, a possible extension of research is how the carrier can use past information to predict the bounds. Since the carrier knows only of his bid in a sealed-bid auction, computing the bounds requires the design of sophisticated algorithms. Naturally, there will be learning effects which influence the performance of the bidding strategies.

Another possible extension is studying our stochastic optimization problem under different probability distributions, e.g. piecewise-uniform or triangular. Furthermore, we may have to estimate the parameters of these distributions with the information revealed over time. Also, in practice, the carriers may bid more aggressively on certain subsets of lanes than others in the hope of winning certain bundles of lanes together. In this case, we have to relax the assumption that the competitors' bids for the lanes are independent. We can no longer compute the probability of winning *exactly* the set S as $P(S, b) \cdot Q(L - S, b)$, where $P(S, b)$ and $Q(L - S, b)$ are given by Equations (3.1) and (3.2).

Notice that our implementation of the coordinate search algorithm is in essence a brute force implementation. As the number of lanes in the instance becomes larger, the brute force implementation is expected to become less and less effective. The number of subsets of L will eventually explode, making the exact evaluation of the objective function difficult. In addition, performing the coordinate search exactly becomes difficult because it also relies on enumeration of the subsets of L which do not include a given lane. Adjustments to the coordinate search algorithm are needed to handle problem instances with a large number of lanes.

A way of dealing with a problem instance with a large number of lanes is decomposing the problem into smaller subproblems and solving each subproblem separately. For instance, the decomposition may be based on the clustering of the lanes in the instance. The subproblems may be solved sequentially one by one on a single processor or they may be solved in parallel on multiple processors.

Another way of dealing with a problem instance with a large number of lanes is approximating the value of the objective function and the coordinate search steps through sampling methods. An important issue here is the sample size to be used. Increasing the sample size results in better approximations but increases computational requirements.

Since the problem we are dealing with is a non-concave maximization problem, an algorithm based purely on coordinate search may not always find the global optimal solution. It may be worthwhile studying global optimization methods. One possibility is a multilevel coordinate search approach proposed by Huyer and Neumaier [38], which is based on iteratively dividing the feasible region into smaller regions and using coordinate search to find locally optimal solutions in these smaller regions. The resulting algorithm is guaranteed to converge if the objective function is continuous in the neighborhood of a global maximizer.

Our assumption that the lanes being auctioned have single period contracts enable us to analyze a given period's auctions independent of the auctions in other periods. If the contracts are for multiple periods while the auctions take place every period, we end up with an auction setting which is a hybrid of sequential and simultaneous auction settings. Under this scenario, the carrier must consider contracts expiring in the subsequent periods and possible contracts arriving in the future periods.

REFERENCES

[1] Albano, G. L., Germano, F., and Lovo, S., "A comparison of standard multi-unit auctions with synergies," *Economics Letters*, vol. 71, pp. 55–60, April 2001.

[2] Alon, N. and Tarsi, M., "Covering multigraphs by simple circuits.," *SIAM Journal on Algorithmic and Discrete Methods*, vol. 6, 1985.

[3] An, N., Elmaghraby, W., and Keskinocak, P., "Bidding strategies and their impact on revenues in combinatorial auctions.," *Journal of Revenue and Pricing Management*, vol. 3, no. 4, pp. 337–357, 2005.

[4] Anandalingam, G., Day, R. W., and Raghavan, S., "The landscape of electronic market design," *Management Science*, vol. 51, no. 3, pp. 316–327, 2005.

[5] BARON: A Global Optimization Software. http://archimedes.scs.uiuc.edu/baron/baron.html, Accessed December 3, 2006.

[6] Benisch, M., Greenwald, A., Grypari, I., Lederman, R., Naroditskiy, V., and Tschantz, M., "Botticelli: A supply chain management agent," in *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, vol. 3, pp. 1174–1181, July 2004.

[7] Bentley, J., "K-d trees for semidynamic point sets," *Proceeding of the 6th Annual ACM Symposium on Computational Geometry*, pp. 187–197, 1990.

[8] Bermond, J., Jackson, B., and Jaeger, F., "Shortest covering of graphs with cycles," *Journal of Combinatorial Theory, Series B*, vol. 35, pp. 297–308, 1983.

[9] Bikhchandani, S., "Auctions of heterogeneous objects," *Games and Economic Behavior*, vol. 26, pp. 193–220, January 1999.

[10] Branco, F., "Sequential auctions with synergies: An example," *Economics Letters*, vol. 54, pp. 159–163, February 1997.

[11] Branco, F., "On the superiority of the multiple round ascending bid auction," *Economics Letters*, vol. 70, pp. 187–194, February 2001.

[12] Bykowsky, M. M., Cull, R. J., and Ledyard, J. O., "Mutually destructive bidding: The fcc auction design problem," *Journal of Regulatory Economics*, vol. 17, pp. 205–28, May 2000.

[13] Candale, T., "Hill-climbing approach to bidding for bundles in simultaneous auctions.," Master's thesis, University of Tulsa, 2005.

[14] Candale, T. and Sen, S., "A comparison of bidding strategies for simultaneous auctions," *ACM SIGecomm Exchange*, vol. 5, pp. 41–48, January 2006.

[15] Caplice, C. and Sheffi, Y., "Optimization-based procurement for transportation services," *Journal of Business Logistics*, vol. 24, no. 2, pp. 109–128, 2003.

[16] Carare, O. and Rothkopf, M., "Slow dutch auctions," *Management Science*, vol. 51, no. 3, pp. 365–373, 2005.

[17] Cassady, R., *Auctions and Auctioneering.* Berkeley: University of California Press, 1967.

[18] Crainic, T. G. and Laporte, G., eds., *Fleet Management and Logistics.* Center for Transportation Research 25th anniversary series, Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[19] Cramton, P., "The fcc spectrum auctions: An early assessment," *Journal of Economics & Management Strategy*, vol. 6, no. 3, pp. 431–495, 1997.

[20] Cybernomics, I., "An experimental comparison of the simultaneous multi-round auctions and the cra combinatorial auctions." Mimeo submitted to the Federal Communications Commission, 2000.

[21] Douma, A., Schuur, P., and van der Heijden, M., "Applying revenue management to agent-based transportation planning.." *Beta Working Paper Series, WP-169*, 2006.

[22] D.Y. Kesel'man, "Covering the edges of a graph by circuits," *Kibernetica*, vol. 3, pp. 16–22, 1987.

[23] Elmaghraby, W., "Pricing and auctions in e-marketplaces," in *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era* (Simchi-Levi, D., Wu, S. D., and Shen, Z. M., eds.), International Series in Operations Research and Management Science, (Norwell, MA), Kluwer Academic Publishers, 2004.

[24] Elmaghraby, W. and Keskinocak, P., "Combinatorial auctions in procurement.," in *The Practice of Supply Chain Management* (Billington, C., Harrison, T., Lee, H., and Neale., J., eds.), (Norwell,MA), Kluwer Academic Publishers, 2003.

[25] Elmaghraby, W. and Keskinocak, P., "Dynamic pricing: Research overview, current practices and future directions.," *Management Science*, vol. 49, no. 10, pp. 1287–1309, 2003.

[26] Engelbrecht-Wiggans, R. and Weber, R. J., "An example of multi-object auction game," *Management Science*, vol. 25, no. 12, pp. 1272–1277, 1979.

[27] Ergun, O., Kuyzu, G., and Savelsbergh, M., "Reducing truckload transportation costs through collaboration." To appear in *Transportation Science*, 2006.

[28] Ergun, O., Kuyzu, G., and Savelsbergh, M., "Shipper collaboration," *Computers & Operations Research*, vol. 34, pp. 1551–1560, July 2007.

[29] Ergun, O. and Ozener, O. O., "Cost allocation in shipper collaborations." Working paper, 2006.

[30] Fan, G., "Covering graphs by cycles," *SIAM Journal on Discrete Mathematics*, vol. 5, pp. 491–496, 1992.

[31] Figliozzi, M., *Performance and Analysis of Spot Truckload Procurement Markets Using Sequential Auctions.* PhD thesis, University of Maryland, 2004.

[32] Foster, J. and Strasser, S., "Carrier/modal selection factors: The shipper/carrier paradox," *Logist. Transportation Rev.*, vol. 31, no. 1, pp. 63–75, 1991.

[33] Fraisse, P., "Cycle covering in bridgeless graphs," *Journal of Combinatorial Theory, Series B*, vol. 39, pp. 146–152, 1985.

[34] Gale, I., "A multiple-object auction with superadditive values," *Economics Letters*, vol. 34, pp. 323–328, December 1990.

[35] Greenwald, A. and Boyan, J., "Bidding under uncertainty: Theory and experiments," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, (Banff, Canada), pp. 209–216, Association for Uncertainty in Artificial Intelligence, 2004.

[36] Guan, M. and Fleischner, H., "On the minimum weighted cycle covering problem for planar graphs," *Ars Combinatoria*, vol. 20, pp. 61–68, 1985.

[37] Hochbaum, D. and Olinick, E., "The bounded cycle cover problem," *INFORMS Journal on Computing*, vol. 13, pp. 104–119, 2001.

[38] Huyer, W. and Neumaier, A., "Global Optimization by Multilevel Coordinate Search," *Journal of Global Optimization*, vol. 14, pp. 331–355, 1999.

[39] Itai, A., Lipton, R., Papadimitrou, C., and Rodeh, M., "Covering graphs by simple circuits," *SIAM Journal of Computing*, vol. 10, pp. 746–750, 1981.

[40] Jackson, B., "Shortest circuit covers and postman tours in graphs with a nowhere zero 4-flows," *SIAM Journal on Computing*, vol. 19, pp. 659–665, 1990.

[41] Katzman, B., "A two stage sequential auction with multi-unit demands," *Journal of Economic Theory*, vol. 86, pp. 77–99, May 1999.

[42] Klemperer, P., "Auctions with almost common values: The 'wallet game' and its applications.," *European Economic Review*, vol. 42, pp. 757–769, 1998.

[43] Krishna, K. and Tranaes, T., "Allocating multiple units," *Economic Theory*, vol. 20, no. 4, pp. 733–750, 2002.

[44] Krishna, V., *Auction Theory.* San Diego, CA: Academic Press, 2002.

[45] Krishna, V. and Rosenthal, R. W., "Simultaneous auctions with synergies," *Games and Economic Behavior*, vol. 17, pp. 1–31, November 1996.

[46] Labbe, M., Laporte, G., and Soriano, P., "Covering a graph with cycles," *Computers and Operations Research*, vol. 25, pp. 499–504, 1998.

[47] Lang, K. and Rosenthal, R. W., "The contractors' game," *RAND Journal of Economics*, vol. 22, pp. 329–338, Autumn 1991.

[48] Ledyard, J. O., Olson, M., Porter, D., Swanson, J. A., and Torma, D. P., "The first use of a combined value auction for transportation services.," *Interfaces*, vol. 32, pp. 4–12, September - October 2002.

[49] Ledyard, J. O., Porter, D., and Rangel, A., "Experiments testing multiobject allocation mechanisms," *Journal of Economics & Management Strategy*, vol. 6, pp. 639–675, 09 1997.

[50] Loftin, J., "Definiteness of a symmetric matrix." http://www.math.columbia.edu/ loftin/ao/polyroot/polyroot.html, Accessed December 3, 2006.

[51] Lucking-Reiley, D., "Using field experiments to test equivalence between auction formats," *Management Science*, vol. 89, no. 5, pp. 1063–1080, 1999.

[52] Lunander, A. and Nilsson, J.-E., "Taking the lab to the field: Experimental tests of alternative mechanisms to procure multiple contracts," *Journal of Regulatory Economics*, vol. 25, pp. 39–58, January 2004.

[53] Maskin, E. and Riley, J., "Asymmetric auctions," *Review of Economic Studies*, vol. 67, pp. 413–438, July 2000.

[54] Mes, M., van der Heijden, M., and Schuur, P., "Opportunity costs calculation in agent-based vehicle routing and scheduling.." *Beta Working Paper Series, WP-168*, 2006.

[55] Milgrom, P., "Game theory and the spectrum auctions," *European Economic Review*, vol. 42, pp. 771–778, May 1998.

[56] Milgrom, P. R. and Weber, P. J., "A theory of auctions and competitive bidding," *Econometrica*, vol. 50, pp. 1089–1122, 1982.

[57] Mishra, D., *Auction Design for Multi-Item Procurement.* PhD thesis, University of Wisconsin, Madison, WI, USA, 2004.

[58] Moore, E. W., Warmke, J. M., and Gorban, L. R., "The indispensable role of management science in centralizing freight operations at reynolds metals company," *Interfaces*, vol. 21, no. 1, pp. 107–129, 1991.

[59] Nandiraju, S. and Regan, A., "Freight transportation electronic marketplaces: A survey of market clearing mechanisms and exploration of important research issues," in *Proceedings 84th Annual Meeting of the Transportation Research Board*, (Washington, D.C.), January 2005.

[60] NEOS Server: BARON. http://neos.mcs.anl.gov/neos/solvers/go:BARON/GAMS.html, Accessed December 3, 2006.

[61] Nistevo, "Collaborative Logistics Network for Transportation Management." http://www.nistevo.com, Accessed December 3, 2006.

[62] Powell, W. B., "Marginal cost pricing of truckload services: A comparison of two approaches.," *Transportation Research, Part B: Methodological*, vol. 19, no. 5, pp. 433–445, 1985.

[63] Powell, W. B., "A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy.," *Transportation Science*, vol. 23, no. 4, pp. 231–243, 1989.

[64] Powell, W. B., Sheffi, Y., Nickerson, K., Butterbaugh, K., and Atherton, S., "Maximizing profits for north american van lines' truckload division: A new framework for pricing and operations.," *Interfaces*, vol. 18, pp. 21–41, 1988.

[65] Rosenthal, R. W. and Wang, R., "Simultaneous auctions with synergies and common values," *Games and Economic Behavior*, vol. 17, pp. 32–55, November 1996.

[66] Rothkopf, M., *Models of auctions and competitive bidding*, vol. 6 of *Handbooks in Operations Research and Management Science*, ch. 19. New York: Elsevier Science Publishing Co., 1994.

[67] Sahinidis, N. V. and Tawarmalani, M., *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2005. Available at http://www.gams.com/dd/docs/solvers/baron.pdf.

[68] Sheffi, Y., "Combinatorial auctions in the procurement of transportation services," *Interfaces*, vol. 34, pp. 245–252, July-August 2004.

[69] Shubik, M., "Auctions, bidding and markets: A historical sketch," in *Auctions, Bidding and Contracting* (Engelbrecht-Wiggans, R., Shubik, M., and Stark, J., eds.), (New York), pp. 33–52, New York University Press, 1983.

[70] Song, J. and Regan, A., "Combinatorial auctions for transportation service procurement: the carrier perspective.," *Transportation Research Record*, vol. 1833, pp. 40–46, 2002.

[71] Song, J. and Regan, A., "Approximation algorithms for the bid valuation and structuring problem in combinatorial auctions for the procurement of freight transportation contracts.," *Transportation Research, Part B: Methodological*, vol. 39, no. 10, pp. 914–933, 2005.

[72] Sydsaeter, K. and Hammond, P. J., *Mathematics for Economic Analysis.* Prentice Hall, 1995.

[73] TAC Team, Wellman, M. P., Wurman, P. R., O'Malley, K., Bangera, R., de Lin, S., and Walsh, D. R. W. E., "Designing the market game for a trading agent competition," *IEEE Internet Computing*, vol. 5, pp. 43–51, March/April 2001.

[74] Tawarmalani, M. and Sahinidis, N. V., "Global optimization of mixed-integer nonlinear programs: A theoretical and computational study," *Mathematical Programming*, vol. 99, no. 3, pp. 563–591, 2004.

[75] Thomassen, C., "On the complexity of finding a minimum cycle cover of a graph," *SIAM Journal on Computing*, vol. 26, pp. 675–677, 1997.

[76] Topaloglu, H. and Powell, W. B., "Sensitivity analysis of a dynamic fleet management model using approximate dynamic programming.," technical report, Cornell University, School of Operations Research and Industrial Engineering, Ithaca, NY, USA, 2004.

[77] Toth, P. and Vigo, D., *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.

[78] Trade Extensions, "Case study: Wooden packaging.." http://www.tradeextensions.com/case-wooden.asp, Accessed October 24, 2006.

[79] Trading Agent Competition. http://www.sics.se/tac/, Accessed November 02, 2006.

[80] Transplace, "The 3PL & Technology Company." http://www.transplace.com, Accessed December 3, 2006.

[81] Vickrey, W., "Counterspeculatoin, auctions, and competitive sealed tenders," *Journal of Finance*, vol. 16, pp. 8–37, 1961.

[82] von Ungern-Sternberg, T., "Swiss auctions," *Economica*, vol. 58, pp. 341–57, August 1991.

[83] W.B. Powell and A. Marar and J. Gelfand and S. Bowers, "Implementing Real-Time Optimization Models: A Case Application From The Motor Carrier Industry," *Operations Research*, vol. 50, pp. 571–581, 2002.

[84] Woolridge, M. and Jennings, N., "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[85] Zeng, D. D., Cox, J. C., and Dror, M., "Coordination of purchasing and bidding activities across markets," in *HICSS '04: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 7*, (Washington, DC, USA), p. 70167.1, IEEE Computer Society, 2004.