# The Pickup and Delivery Problem with Split Loads

A Thesis
Presented to
The Academic Faculty

by

**Maciek A. Nowak**

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Industrial and Systems Engineering
Georgia Institute of Technology
August 2005

**The Pickup and Delivery Problem with Split Loads**

Approved by:

Professor Chelsea C. White III
School of Industrial and Systems Engineering
College of Engineering, Committee Chair

Professor Bruce Ellingwood
School of Civil and Environmental Engineering
College of Engineering

Professor Ozlem Ergun
School of Industrial and Systems Engineering
College of Engineering, Adviser

Professor Joel Sokol
School of Industrial and Systems Engineering
College of Engineering

Professor Martin Savelsbergh
School of Industrial and Systems Engineering
College of Engineering

Date Approved 5/17/05

# ACKNOWLEDGEMENTS

While working on my Ph.D. at Georgia Tech and Michigan, I have received a great deal of support from many different people. My degree would not have been possible without this support. I would like to begin by thanking my two dissertation advisors, Dr. Ozlem Ergun and Dr. Chip White. Their guidance and encouragement was constant and unwavering. They helped open the door for me to countless new intellectual discoveries.

While completing my Ph.D., I spent one year working as an intern for Federal Express. I am very grateful for the opportunity to gain this real-world experience. Particularly, I would like to thank Mike Herringshaw, Dave Schwartz, Scott Gardner and Aaron Meadows for the time they devoted to working with me.

I also would like to thank the faculty at both Michigan and Georgia Tech. While the faculty at Michigan helped to develop the framework for my education, the Georgia Tech faculty were able to answer the many questions that came up in writing my dissertation. In particular, I would like to thank Dr. Steve Pollock, Dr. Marina Epelman, Dr. Larry Seiford, and Dr. Mark Lewis at Michigan and Dr. Faiz Al-Khayyal, Dr. Alan Erera and Dr. Anton Kleywegt at Georgia Tech. I would also like to thank Dr. Martin Savelsbergh, Dr. Joel Sokol and Dr. Bruce Ellingwood for serving on my dissertation committee and always being there to provide me with much needed advice.

I also owe a great deal of gratitude to all the friends who helped me out through this process. They were always willing to take a break from their own work to help me out with any problem, from how to finish a proof to how to finish a fifth. There are too many people to mention here, but any list would have to include the IOE crew, who showed me that the graduate years can be more fun than the undergraduate, Brian and Lori, who helped me figure Tech out, Rahul and Toby, who may have inadvertently delayed the completion of my dissertation during "the stretch," and my best friend, Roy, the only other person I've ever met who would be willing to start cooking a chicken at three in the morning because

he's hungry.

Finally, I would like to thank my family. Without their unquestioning support I never would have finished. Thanks to Dad for his step-by-step advice on the whole process, to Mom for her regular care packages, and to Dorotka and Kasia for convincing me that I was smarter than I really am.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

This dissertation focuses on improvements in vehicle routing that can be gained by allowing multiple vehicles to service a common load. We explore how costs can be reduced through the elimination of the constraint that a load must be serviced by only one vehicle. Specifically, we look at the problem of routing vehicles to service loads that have distinct origins and destinations, with no constraint on the amount of a load that a vehicle may service. We call this the Pickup and Delivery Problem with Split Loads (PDPSL). We model this problem as a dynamic program and introduce structural results that can help practitioners implement the use of split loads, including the definition of an upper bound on the benefit of split loads. This bound indicates that the routing cost can be reduced by at most one half when split loads are allowed. Furthermore, the most benefit occurs when load sizes are just above one half of vehicle capacity.

We develop a heuristic for the solution of large scale problems, and apply this heuristic to randomly generated data sets. Various load sizes are tested, with the experimental results supporting the finding that most benefit with split loads occurs for load sizes just above one half vehicle capacity. Also, the average benefit of split loads is found to range from 6 to 7% for most data sets. The heuristic was also tested on a real world example from the trucking industry. These tests reveal the benefit of both using split loads and allowing fleet sharing. The benefit for split loads is not as significant as with the random data, and the various business rules added for this case are tested to find those that have the most impact. It is found that an additional cost for every stop the vehicle makes strictly limits the potential for benefit from split loads. Finally, we present a simplified version of the PDPSL in which all origins are visited prior to any destination on a route, generalizing structural results from the Split Delivery Vehicle Routing Problem for this problem.

# CHAPTER I

# INTRODUCTION

The continuing surge in oil prices has led many companies to analyze their transportation operations more meticulously than ever before. As trucking accounts for the overwhelming majority of transportation expenditures relative to other modes, it has received the most attention. Various methods have been utilized to conserve costs associated with trucking operations, including the implementation of information technology, new uses of vehicle technology, and more effective scheduling methods. One of the most commonly applied techniques to improve operations in trucking is the implementation of vehicle routing algorithms to more efficiently take advantage of vehicle time and capacity. By effectively routing a vehicle and selecting the loads to be placed on the vehicle, a company may significantly reduce transportation costs.

This dissertation focuses on improvements in vehicle routing that can be gained by allowing multiple vehicles to service a common load. We explore how costs can be reduced through the elimination of the constraint that a load must be serviced by only one vehicle. Specifically, we look at the problem of routing vehicles to service loads that have distinct origins and destinations, with no constraint on the amount of a load that a vehicle may service. We call this the Pickup and Delivery Problem with Split Loads (PDPSL). We model this problem as a dynamic program and introduce structural results that can help practitioners implement the use of split loads. We also quantify the benefit of using split loads and describe those instances that allow for the most benefit.

## *1.1 Overview*

As cost saving measures are explored throughout various industries, transportation is one of the first areas expected to make sacrifices. One of the main reasons for this is that transportation can be a glaringly large portion of a company's budget, with little actual

revenue generation. Trucking has received the most attention, as it accounts for the most significant portion of transportation spending. In 2003, the U.S. trucking industry generated revenues of over $530 billion, accounting for 87% by value, and 68.9% by tonnage, of all freight shipped in the United States [2]. While many companies struggle to cap transportation costs, trucking firms are pushed to offer more services at reduced rates. While external pressure grows, competition within the industry is also building, with the number of firms continually increasing [26]. Trucking companies are operating on margins of only 3 to 4 cents on the dollar in some instances [57]. However, despite the various cost saving measures implemented, transportation continues to suffer from inefficiency. A major indicator of this inefficiency is the common occurrence of excess capacity within a firm's fleet of vehicles. Frequent partial loads or empty backhauls reveal that many vehicles often travel with significant amounts of available capacity. Every one of these occurences represents a potential for cost savings.

An obvious solution to the problem of excess capacity is to pool partial (or full) loads traveling similar routes onto common vehicles. If a supplier is providing parts for two manufacturers in the same vicinity, using one vehicle for both deliveries is clearly more efficient. Similarly, a vehicle returning on an empty backhaul may be used to transport a load intended to travel close to its return route. This improvement in efficiency can occur with the enhancement of communication between separate entities with similar transportation needs. If two closely located manufacturers share information regarding transportation needs, there is a possibility that they could share the costs of those needs. These opportunities are currently being made use of through a variety of avenues, including internet transportation exchanges and the pooling of transportation resources within certain industries. However, the occurrence of excess capacity continues, somewhat unabated. For a variety of reasons, many relationships that could be exploited for mutual benefit are not. An initial test study on enhancing communication for paratransit service providers in Ann Arbor, MI showed a 15% increase in the number of passengers serviced per hour and a 6% cost savings, through the use of a very simple routing procedure. [71]

The benefits of allowing for the pooling of loads with fleet sharing are easy to see.

However, a more subtle extension of this option is to allow not only any vehicle to serve any load, but also multiple vehicles to service the same load. This can be further extended to allow a vehicle to service the same load multiple times on one route. Splitting a load such that the delivery of that load is completed in pieces rather than all at once appears to initially generate an additional cost for each trip that services a part of the load. However, there are numerous instances in which load splitting results in a cost reduction. Several studies have shown the benefit of split deliveries for the Vehicle Routing Problem (VRP), in which a vehicle operating out of a depot makes a series of deliveries on each route ([20], [21], [22], [29]). In this dissertation, we quantify the benefit of using split loads for the Pickup and Delivery Problem (PDP), in which each load has a specific origin and destination associated with it. Figure 1 presents an example of the potential benefit. The load sizes represent the portion of a truckload (TL) that is to be delivered. In this example, all three loads share a common destination. Without split loads, the vehicle must deliver each of the loads individually, as capacity constraints prohibit the combination of any loads on the vehicle. With split loads, the vehicle may pick up load $A$, split load $B$ by picking up 0.4 truckloads, and deliver both to the destination. It may then return for the other half of load $B$, pick up load $C$, and deliver both to the destination. The benefit of using split loads is then the elimination of two of the long paths between the origins and destination (with the addition of two short trips between the origins).

The following section introduces the Pickup and Delivery Problem with Split Loads (PDPSL).

## *1.2 Description of the Pickup and Delivery Problem with Split Loads*

The Pickup and Delivery Problem with Split Loads (PDPSL) is motivated by the less-than-truckload (LTL) trucking industry and the effect that the use of split loads could have on its operation. The LTL trucking industry, in contrast to the truckload trucking industry, is the portion of the industry that moves customer loads that do not fill an entire trailer. Traditionally, these shipments weigh between 150 and 10,000 pounds [7]. In general, a load

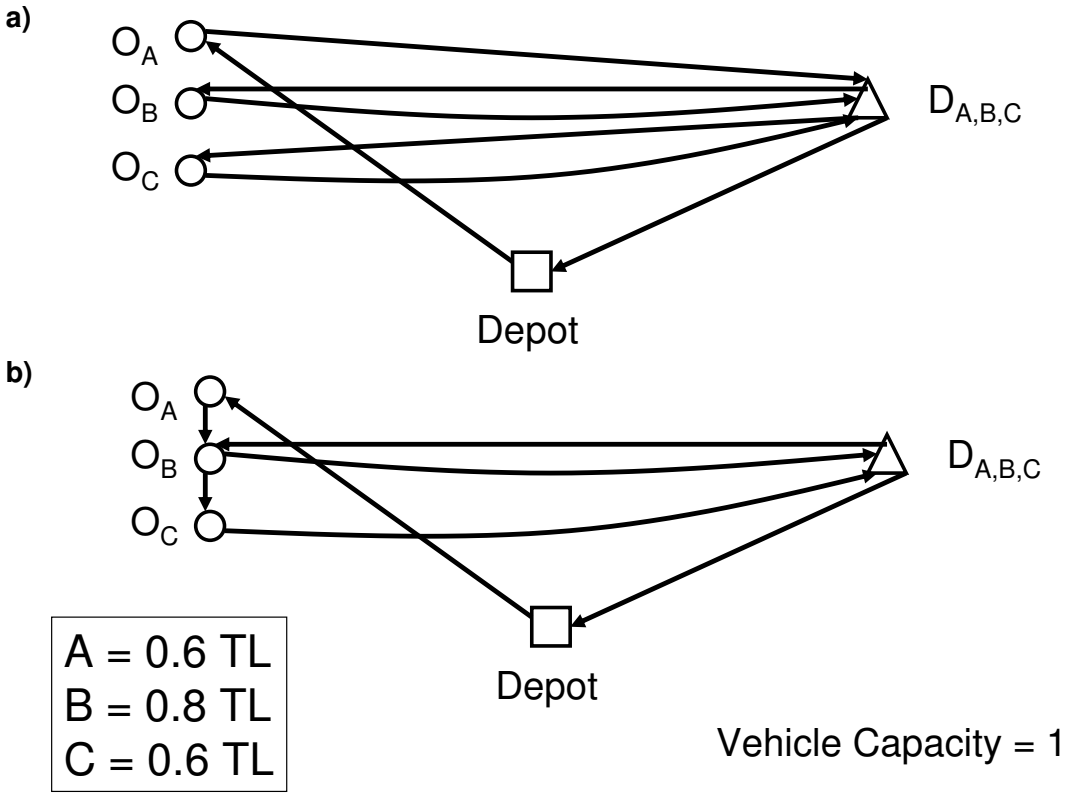


**Figure 1:** Example of benefit of split loads showing a) route plan without split loads; and b) route plan with split loads.

is picked up at an origin and delivered to some destination on a vehicle route. These pickups and deliveries may occur in any order, as long as vehicle capacity is not violated and the origin of a load is visited prior its destination. The vehicle begins and ends the day empty, at a depot.

Consider a fleet of capacitated vehicles, all with the same capacity, that travel on a road network with all distances known. Each vehicle accrues a travel cost each time it travels a road segment or arc in the network, and this cost is arc dependent. We assume that the costs associated with each arc are symmetric and the triangular inequality is obeyed. The vehicles must service a set of loads to be picked up from their respective origins and delivered to their destinations. The problem objective is to determine a policy for selecting the next node for a vehicle to visit and, if the vehicle arrives at an origin, the amount of a load to pick up, that minimizes the total cost of each vehicle trip.

Consider a single vehicle with some set of loads to be picked up and delivered. The path of the vehicle is such that it leaves the depot empty and must first visit an origin (a destination will not be visited as the vehicle is empty). Either the entire load located at the origin, or some portion of the load, is picked up. The vehicle will not visit an origin if it does not pick up a load at that location. If loads to be dropped off at multiple destinations are located at the origin, the vehicle may pick up several loads. Vehicle capacity must not be violated in selecting any loads for pick up. After visiting the initial origin, the vehicle may either visit another origin or travel to a destination. If another origin is visited, an additional load(s) must be picked up, abiding by capacity constraints. If a destination is visited, all loads on the vehicle destined for that location are removed from the vehicle. If the vehicle is empty after visiting a destination, it may either continue to another origin, or return to the depot. If the vehicle is not empty, it may travel to an origin or another destination.

The decision made at each origin involves the selection of a load amount to place on the vehicle. As indicated earlier, the size of the load picked up may be the entire load, or some fractional portion of the load. We now define the two conditions under which a load is considered to be split:

1. when the total load size to be delivered between a specific origin and destination is less than or equal to vehicle capacity, a load is split if the vehicle picks up any amount that is less than the full load size requiring service.

2. when the load size to be delivered is greater than vehicle capacity, a load is split if the number of pickups used to service the load is greater than the upper integer of the total load size divided by vehicle capacity. For example, if a load is 3.6 truckloads and vehicle capacity is 1, at least four trips will be required to fully service the load. If the load is picked up in five or more increments, than it is considered to be split.

The classification of what is considered to be a split does not affect the resulting solution of the PDPSL. Rather, it is beneficial when determining the number of split loads per route, particularly when limiting the number of splits allowed.

## 1.3 Dissertation Outline

In this section, we provide an outline of this dissertation. The remainder of this chapter is devoted to discussing the literature relevant to the problems introduced in this dissertation.

Chapter 2 presents the basic concepts of the Pickup and Delivery Problem with Split Loads. Section 2.1 formulates the PDPSL as a dynamic program. Section 2.2 shows that the solution cost of the problem is monotonically increasing with the size of loads to be delivered and Section 2.3 presents an upper bound on the benefit of the use of split loads. Section 2.4 describes an alternative exact formulation, consisting of a nonlinear program that is converted into a mixed integer program.

Chapter 3 presents a heuristic that uses elements of a "classical" construction heuristic and newer "metaheuristic" methods to solve the PDPSL. Section 3.1 provides a background on the methods applied to the heuristic described in Section 3.2. Section 3.3 expands on this description with a specific framework for the heuristic.

Chapter 4 uses the heuristic described in Chapter 3 to show the benefit of split loads. In Section 4.1 the design of the experiment used to evaluate split loads is described, while Section 4.2 presents the results of the experiment. Section 4.3 provides a benchmark for the heuristic by comparing it to the mixed integer program described in Section 2.4.

Chapter 5 is a case study in which the heuristic is applied to data provided by Federal Express. Section 5.1 discusses the historical background of Federal Express and indicates the motivation for this case study. Section 5.2 describes the problem to be analyzed, while Section 5.3 presents the design of the experiment used for this analysis. Sections 5.4 and 5.5 present and interpret the various results of the analysis.

Chapter 6 follows an outline similar to Chapter 2, with a description of the Constrained Pickup and Delivery Problem with Split Loads. An additional constraint is added to the PDPSL, requiring that all origins must be visited before any destinations on each route. Section 6.1 provides the dynamic programming formulation for the revised problem with the additional constraints. Section 6.2 presents several important properties of the CPDPSL that are generalized from the Split Delivery Vehicle Routing Problem. Section 6.3 explains how the action space may be reduced, while Section 6.4 discusses the use of the reduced

action space in creating a finite state space. The A* algorithm used for the solution of the CPDPSL is described in Section 6.5. Sections 6.6 and 6.7 present the design and results of the experiment used for analysis.

Chapter 7 discusses future work suggested by the research presented in this dissertation.

## *1.4 Literature Review*

In this section, we present a review of the literature pertinent to the topic of the Pickup and Delivery Problem with Split Loads. The two main bodies of routing literature that are relevant to this problem cover the Vehicle Routing Problem (VRP) and an extension of the VRP, the Pickup and Delivery Problem (PDP). The latter problem is more relevant to the work presented here; however, most work with split loads has been done in connection with the VRP. We were not able to find references that allowed split loads with the PDP. In addition to the work that applies split loads to the VRP, known as the Split Delivery Vehicle Routing Problem (SDVRP), several other extensions of this problem have characteristics that may be of interest. The Vehicle Routing Problem with Backhauls (VRPB) is similar to the PDP, in that both pickups and deliveries may be made by the vehicle at separate origins and destinations. Similarly, the Vehicle Routing Problem with Simultaneous Delivery and Pickup (VRPSDP) allows a vehicle to perform both pickups and deliveries, but at the same location.

We first describe the VRP literature, the direct extensions to the problem that are pertinent to this dissertation, and the several relevant solution techniques that are currently available. This is followed by the PDP literature, with a review of both exact and approximate solution methods.

### 1.4.1 Vehicle Routing Problem

The Vehicle Routing Problem is a direct extension of the well-known Traveling Salesman Problem (TSP). The TSP is perhaps the most widely used routing problem, with early work that can be traced to the nineteenth century mathematician Sir William Hamilton and his Hamiltonian cycle, in which each node on a graph is visited exactly once. However, the TSP has seen most development over the last fifty years, beginning with Dantzig, Fulkerson

and Johnson [14]. A thorough review and survey of the TSP literature is found in Lawler et al. [56], and discussions of algorithms used for the TSP can be found in Nemhauser and Wolsey [69] and Gutin and Punnen [43].

The VRP extends the TSP to a set of vehicles serving a set of customers. Each customer requires some level of service, which can be thought of as demand, and the vehicles have a limited capacity with which to service the demand. Like the TSP, the VRP has been extensively studied. Golden and Assad [42] have edited a book detailing this subject, while Laporte and Osman [54] present an extensive bibliography. Toth and Vigo [94] present a review of exact solution methods for the VRP. However, we are more interested in approximate methods used for the VRP, and a brief review of this literature follows.

#### *1.4.1.1 Heuristic Methods for the Vehicle Routing Problem*

The complexity of the Vehicle Routing Problem (VRP), and the more general Pickup and Delivery Problem, precludes the exact solution of any realistically sized problem in a reasonable length of time. Therefore, researchers have turned to heuristics for help in approximating good solutions. Cordeau et al. [13] provide a thorough review of various VRP heuristics. There is a significant body of literature touching on solution techniques that apply to all variants of the VRP. This review focuses primarily on local search methods, particularly those used with the tabu search heuristic. These methods are most pertinent to the work presented here. A discussion of heuristic methods can be found in Chapter 3.

Gendreau et al. [32] applied the tabu search heuristic to the VRP with their Taburoute algorithm. Single insertion moves defined the neighborhood of the solution, with the possibility that infeasible solutions may be explored. Taillard's [90] tabu search algorithm used both insertion and swap moves, analyzing only feasible routes. Xu and Kelly [98] evaluate ejection, insertion and swap moves in a local search approach based on a network flow model, also incorporating tabu search. Granular tabu search is introduced by Toth and Vigo [95], in which the solution neighborhood is drastically restricted by eliminating moves that are not likely to lead to good solutions. Prins [75] presents a simple hybrid genetic algorithm that is very competitive with the tabu search heuristic methods discussed above.

#### *1.4.1.2 Split Delivery Vehicle Routing Problem*

The use of split loads has been most closely studied when applied to the Vehicle Routing Problem. This special case is known as the Split Delivery Vehicle Routing Problem (SDVRP), and occurs when a destination may be serviced by multiple vehicles, resulting in split deliveries. While the PDPSL is a more complex problem than the SDVRP, the approaches used for solving the SDVRP are applicable and deserve to be studied. In the special case of the PDPSL detailed in Chapter 6, in which all pickups must precede any deliveries on a route, most of the findings regarding the SDVRP directly apply and can be translated to the PDPSL.

The SDVRP was first introduced by Dror and Trudeau [21], where they found instances of almost 14% cost benefit from the use of split deliveries for certain problem sets. These savings were found through a heuristic that generates splits based on the cost savings of removing a delivery from one route and dividing it between two other routes (the split generation was built on a routing heuristic for the general VRP). Dror and Trudeau [22] later formally formulated the problem as an integer linear program and further developed their heuristic procedure. They also presented an interesting result that indicates that no two routes in the optimal solution can have more than one split demand point in common. This finding was extended by Lee et al. [58] to show that there exists an optimal solution in which no route contains more than one split delivery. Dror and Trudeau [20] continue their work on the SDVRP, defining several new classes of valid constraints to reduce the gap between the lower and upper bounds at the root of the search tree used in their branch and bound algorithm.

Frizzell and Giffin [29] propose a construction heuristic that accounts for grid network distances, uses bounds that limit the ways in which a delivery may be split and incorporates the costs of splitting that may be accrued by a customer. Following Dror and Trudeau's results, the heuristic produces split delivery solutions that outperform those solutions where splits are prohibited. This work was later extended to account for time windows, with a construction heuristic that assigned customers based on time remaining before the end of a customer's time window [30]. Federgruen and Simchi-Levi [25] treat a demand $d$ as $d$ clients

with unit size demand, all at the same location. With this assumption, they analyze the performance of several heuristic procedures. The polyhedron of the SDVRP is examined by Belenguer et al. [9], with the insights leading to the development of a new set of valid inequalities and a lower bound that allowed for the optimal solution of instances with up to 50 customers. Ho and Haugland [46] present a tabu search heuristic that applies four common move operators while simultaneously generating split loads. The split loads are created based on the amount of available capacity on a route when a load is to be placed on the route. Archetti et al. [5] also apply the tabu search heuristic to the SDVRP. Insertion moves are used for local improvement, with the possibility of inserting a customer into a route without removing it from another route.

Mullaseril et al. [66] apply the SDVRP to the problem of distributing feed on a cattle ranch. The problem is treated as a Capacitated Rural Postman Problem, in that the demands are found on arcs instead of nodes, with time windows. Heuristic techniques similar to those used by Dror and Trudeau [22] are applied here, with comparable results. In another application of this problem, Sierksma and Tijssen [89] present a solution to the SDVRP used to route helicopters for crew exchanges on off-shore drilling locations in the North Sea. They develop both an exact solution procedure that solves a linear programming model through column generation and a Cluster-and-Route heuristic that utilizes both 1-opt and 2-opt procedures.

#### *1.4.1.3 Vehicle Routing Problem with Backhauls*

An extension of the VRP that is similar to the PDP is the Vehicle Routing Problem with Backhauls (VRPB), in which there are two subsets of customers known as linehauls and backhauls. Linehaul customers require delivery of goods from some depot, while backhaul customers have goods to be picked up and brought to the depot. The goods are transported to or from a single depot by a fleet of homogeneous vehicles. The VRPB primarily differs from the PDP in that the backhaul, or pick up, customers are not necessarily linked to the linehaul, or delivery, customers, by a load that must be picked up at one customer and dropped off at another. That is, in the VRPB, all loads originate or end at the depot, while

in the PDP loads are transferred from one customer to another. Therefore, a vehicle leaves from and returns to the depot empty in the PDP, while it returns to the depot with loads picked up from backhaul customers in the VRPB. However, despite these differences, we present some of the methods used for solving this problem, in particular the insertion and tabu search heuristics, as they are applicable to this paper.

Casco et al. [10] provide an overview of the VRPB and present some initial algorithmic techniques used to analyze the problem. Research in this area has focused on both instances in which all linehauls must be visited before any backhauls on a route and those instances when linehauls and backhauls may be mixed. Duhamel et al. [23] use a tabu search heuristic to compare the two options, finding benefit both when backhauls are allowed in general and when they are mixed with linehauls. Similarly, Wade and Salhi [97] use a simple insertion heuristic to find the optimal mix of routing linehaul and backhaul customers. However, the majority of literature in this area focuses on one of the two instances and can be divided into exact algorithms and heuristic methods.

The first heuristic to solve the VRPB was proposed by Deif and Bodin [15], in which the Clarke and Wright Savings Algorithm was extended by modifying the savings of the arcs connecting linehaul and backhaul customers, effectively delaying the formation of routes that mix the two types of customers. Goetschalckx and Jacobs-Blecha [40] presented an algorithm that uses spacefilling curves, originally proposed by Bartholdi and Platzman [8], to generate initial routes for customers of the same type. The routes were then combined to form the final set of vehicle routes. Goetschalckx and Jacobs-Blecha [41] further extended this work, based on the Fisher-Jaikumar [28] VRP heuristic, through an adaptation of the Generalized Assignment heuristic. A cluster-first-route-second heuristic was developed by Toth and Vigo [93], with computational results indicating that they outperform the earlier heuristics developed by Goetschalckx and Jacobs-Blecha [41]. Osman and Wassan [72] were the first to apply tabu search to the VRPB, with initial solutions created through a saving-insertion heuristic and a saving-assignment heuristic.

Several heuristics focus on those instances where the linehaul and backhaul customers may be mixed. Salhi and Nagy [83] make improvements on cluster heuristics by inserting

multiple backhauls into a route at the same time, with no additional computational burden. A simplified version of the VRPB with mixed loads, in which there is a single vehicle, known as the Traveling Salesman Problem with Backhauls (TSPB) is presented by Mosheiov [65]. Methods used for the standard TSP are extended and it is shown that if a solution is infeasible because an arc is overloaded, feasibility may be restored by inserting the depot into the arc. A minimum spanning tree for the TSPB is presented by Anily and Mosheiov [4], with an improvement in computational complexity relative to the model presented by Mosheiov [65], but a decrease in average performance. Gendreau et al. [33, 34] also present heuristics for the TSPB, but without mixing between linehaul and backhaul customers. In an extension of the VRPB, several heuristics have implemented time windows, as in Cheung and Hang [11], Duhamel et al. [23], Potvin et al. [74] and Gèlinas et al. [31]; however, this work is not applicable to the work presented here.

Exact solution methods are limited in the literature, as large-sized problems cannot be solved efficiently by these procedures, and the focus is on those instances with a precedence constraint for linehaul customers. Yano et al. [99] were the first to provide an exact algorithm, in which the number of customers of each type in a route is not greater than 4. Toth and Vigo [91] present an integer linear programming model and an effective overall bounding procedure that utilizes a Lagrangian lower bound. Mingozzi et al. [62] present an alternate integer programming formulation and use several heuristic methods to find a valid lower bound.

#### *1.4.1.4 Vehicle Routing Problem with Simultaneous Delivery and Pickup*

Closely related to the VRPB, the Vehicle Routing Problem with Simultaneous Delivery and Pickup (VRPSDP) is a generalization that allows for both delivery and pickup at the same location. This problem has become more important as the area of reverse logistics has grown in popularity, and concerns rise over recycling or returning packaging. Some argue that the VRPSDP may be solved using a special case of the VRPB in which each customer is treated as two customers, a "delivery-only" and "pickup-only." This is not necessarily accurate, as under the VRPB, backhauls are often visited after all linehauls have been completed, and

customers may be visited twice. The goal of the VRPSDP, as the name suggests, is to perform the pickup and delivery simultaneously, so as to minimize inconvenience to the customer.

While the literature is more sparse in this area, it is important to present this problem as the additional constraint relates to the work presented here. Many customers in the Pickup and Delivery Problem must be treated as both an origin and a destination. Also, as in the PDP, the utilization of vehicle capacity is dependent not only on the assignment of deliveries/pickups to the vehicle, but on the order in which those deliveries/pickups are made on the route as well. This also holds for the VRPB in which linehauls and backhauls may be mixed, but to a lesser extent.

Not surprisingly, the VRPSDP literature builds upon much of the work discussed in the section on the VRPB. This problem was first introduced by Min [61], where customers were first clustered into groups and then ordered by solving the traveling salesman problem on each group. Halse [44] proposed a heuristic algorithm based on a Lagrangian relaxation of the problem, again first clustering the customers into groups and then using a routing procedure based on 3-opt. However, this problem has gained more attention only recently. Dethloff [18] first discusses the applicability of VRPB solution methods for the VRPSDP, finding that those solutions may result in a lower cost, but with a violation of the one stop per customer constraint. An insertion heuristic that adds nodes to a route based on remaining vehicle capacity is presented, which Dethloff [19] then compares to a VRPB heuristic. As expected, the VRPSDP heuristic performs favorably when the number of backhauls requested is equal to the number of linehauls, as there is more opportunity to take advantage of the tool that monitors vehicle capacity. Angelelli and Mansini [3] present an exact formulation for the problem, using a Branch and Price approach based on a set covering formulation. Similarly, Dell'Amico et al. [16] present an optimization algorithm based on column generation, dynamic programming and Branch and Price. Weak and strong feasibilities are separated in a heuristic developed by Nagy and Salhi [67] that uses several improvement techniques similar to those used in this work. In addition to a route combination procedure, 2-opt, 3-opt and customer exchange routines are utilized for the

VRPSDP. As with the VRPB, a simplified version of the VRPSDP in which there is a single vehicle is presented by Gendreau et al. [36].

### 1.4.2 Pickup and Delivery Problem

The PDP has been extensively studied, with Savelsbergh and Sol [84] providing a thorough review of this work. The most commonly studied variant of the PDP is the Pickup and Delivery Problem with Time Windows (PDPTW). A survey of this problem is presented by Mitrović-Minić [63]. Although the focus of this paper is on the PDP without time windows, most heuristics take time windows into account. Those with techniques applicable or related to the work presented here are discussed below.

#### *1.4.2.1 Optimization Methods for Pickup and Delivery Problem*

The primary focus of this work is on heuristic procedures for the PDPSL. However, the body of work on the optimization methods for the PDP is helpful, particularly in providing a benchmark to evaluate heuristic methods. The first exact solution method for the PDP was developed by Psaraftis [76]. A dynamic programming approach is developed, with solutions presented for up to 9 customers. Kalantari et al. [51] extend the branch and bound algorithm developed by Little et al. [59] to the PDP, dividing tours into manageable subsets to optimally solve for problem instances with up to 31 customers. An additive bounding approach is applied in a branch-and-bound algorithm by Fischetti and Toth [27], with bounds based on assignment relaxation, the shortest spanning 1-arborescence relaxation, disjunctions and variable decomposition. Ruland and Rodin [82] present an integer programming formulation that is solved through the use of a branch-and-cut algorithm. Lu and Dessouky [60] also present a branch-and-cut algorithm, based on a 0-1 integer programming formulation. Exact solution methods of the PDPTW include Psaraftis' [78] modification of the dynamic programming algorithm applied in [76], an alternative dynamic programming formulation presented by Desrosiers et al. [17] and a set partitioning formulation solved by column generation presented by Dumas et al. [24]. Desrosiers et al. [17] used multiple criteria based on precedence constraints and time windows to reduce the state space, a technique similar to that presented in Chapter 6.

#### *1.4.2.2 Heuristic Methods for Pickup and Delivery Problem*

The PDP is a generalization of the Traveling Salesman Problem(TSP), making it NP-hard in the strong sense, focusing most current research on heuristic methods. A worst case analysis of a two-phase approximation algorithm is presented by Psaraftis [77], with initial TSP tour construction in the first phase and improvements made while traversing the tour in the second phase. This algorithm was improved by Renaud et al. [81], with the implementation of a deletion and re-insertion heuristic in the second phase.

The heuristics for the PDP described above did not take into account time windows; however, the larger body of work on the PDP does. Van der Bruggen et al. [96] use a two-phase local search algorithm for the PDPTW similar to those discussed for the PDP. Both phases use a variable-depth procedure and an embedded arc-exchange algortihm. Sexton and Bodin [86, 87] applied Benders' decomposition procedure to a mixed binary nonlinear formulation that solves the scheduling and routing components of the PDPTW separately. Sexton and Choi [88] extend this work to a version of the problem with split loads; however, almost no focus was placed on this characteristic of the problem. Rather, the primary goal was to minimize a linear combination of total vehicle operating time and total customer penalty due to missing any of the time windows. Jaw et al. [50] present an insertion heuristic that generates routes by selecting insertions based on feasibility and cost. A new local search extension called sacrificing is introduced by Healy and Moll [45], finding significant improvements in solution quality with no deterioration of run time. Savelsbergh and Sol [85] apply a sophisticated column management scheme to a branch-and-price algorithm in the development of a decision support system for a European road transportation firm. Three different perturbation heuristics are used by Renaud et al. [80] to help a local search process move away from a local optimum.

The tabu search heuristic developed by Nanry and Barnes [68] is of some interest due to the move neighborhoods used in this work. In order to generate solutions, origin and destination locations may be either inserted from one route to another or swapped with an origin-destination pair on another route. These moves are similar to those applied in the heuristic presented in this paper. However, vehicle capacity constraints may be violated by

these moves, with a third type of move inserting individual origins or destinations forward or backward within a route to improve the solution quality or feasibility. Similarly, Toth and Vigo [92] presented a parallel insertion heuristic that uses tabu thresholding with several moves like those used in this work. Landrieu et al. [53] also present a tabu search heuristic that uses local improvements. These moves are simpler than those proposed by Nanry and Barnes [68] or those used in this work, as vertices are swapped or inserted within a route, while respecting precedence and vehicle capacity constraints.

## *1.5 Contribution*

The primary contribution of this dissertation is the introduction of the Pickup and Delivery Problem with Split Loads into the vehicle routing literature. As indicated previously, to the best of the author's knowledge, the PDPSL has not been studied in the literature. For this problem, we introduce a dynamic programming model and identify several structural results, including the definition of an upper bound on the benefit of split loads. This bound indicates that the routing cost can be reduced by at most one half when split loads are allowed. Furthermore, the most benefit occurs when load sizes are just above one half of vehicle capacity.

We develop a heuristic for the solution of large scale problems, and apply this heuristic to randomly generated data sets. Various load sizes are tested, with the experimental results supporting the finding that most benefit with split loads occurs for load sizes just above one half vehicle capacity. Also, the average benefit of split loads is found to range from 6 to 7% for most data sets. The heuristic was also tested on a real world example from the trucking industry. These tests reveal the benefit of both using split loads and allowing fleet sharing. The benefit for split loads is not as significant as with the random data, and the various business rules added for this case are tested to find those that have the most impact. It is found that an additional cost for every stop the vehicle makes strictly limits the potential for benefit from split loads. Finally, we present a simplified version of the PDPSL in which all origins are visited prior to any destination on a route, generalizing structural results from the Split Delivery Vehicle Routing Problem for this problem.

# CHAPTER II

# EXACT FORMULATIONS AND STRUCTURAL RESULTS

In this chapter we introduce the Pickup and Delivery Problem with Split Loads (PDPSL), presenting two exact formulations and structural results that stem from those. Section 2.1 formulates the PDPSL as a dynamic program. Section 2.2 shows that the solution cost of the problem is monotonically increasing with the size of loads to be delivered and Section 2.3 presents an upper bound on the benefit of the use of split loads. Section 2.4 describes an alternative exact formulation, consisting of a nonlinear program that is converted into a mixed integer program. Section 2.5 provides conclusions from the chapter.

## *2.1 Model Formulation*

Let the graph $G = (N, E)$ serve as a model of a fully connected road network, where $N$ is the finite set of nodes, modeling intersections, and $E \subseteq N \times N$ is the set of directed arcs modeling one-way roads between intersections. That is, $(n, n') \in E$ if and only if there is a road that permits traffic to flow from intersection $n$ to intersection $n'$. Let $SCS(n) = \{n' : (n, n') \in E\}$ be the successor set of node $n$. We assume throughout that $n \notin SCS(n)$; thus, idling at an intersection is not permitted.

Pickups can be made at the origin nodes, $O \subset N$, and deliveries can be made at the destination nodes, $D \subset N$. We remark that some nodes may fall in both sets, such that $O \cap D \neq 0$; that is, there may be nodes that serve as both origin and destination nodes (e.g., a crossdock). There also exists a node $Depot \subset N$, where $Depot \notin O \cap D$. The vehicle leaves from, and returns to, the $Depot$ empty at both the beginning and the end of a route. Because the vehicle starts from and returns to the depot, each route may be serviced by a separate vehicle. However, for simplicity of analysis, we assume that there is a single pickup and delivery vehicle. A route is a sequence of nodes $(Depot, n_1, ..., n_Q, Depot)$,

with $n_q \in N$ for all $q = 1, ..., Q$, and $n_{q+1} \in SCS(n_q)$ for all $q = 1, ..., Q-1$. We assume $O \cup D \cup \{Depot\} = N$.

Let $a(i,j)$ be the amount of goods stored at $i \in O$ destined for $j \in D$, and let $a = \{a(i,j) : (i,j) \in O \times D\}$. Let $b(j)$ be the amount of goods currently on the vehicle destined for $j \in D$, and let $b = \{b(j) : j \in D\}$. The pair $(a,b)$, called the inventory position, is said to be feasible ($(a,b) \in \mathcal{F}$) if and only if

i. $0 \leq a(i,j)$ for all $(i,j) \in O \times D$

ii. $0 \leq b(j)$ for all $j \in D$

iii. $\sum_{j \in D} b(i,j) \leq 1$

where without loss of generality we assume that the capacity of the vehicle is 1.

A decision epoch occurs when the vehicle arrives at any node in $N$. Let $t_q$ be the time of the $q$th decision epoch, and let $n_q \in N$ be the position of the vehicle at time $t_q$. Let $T$ be a finite integer indicating that action selection terminates at a time no later than $T-1$, and $Q$ is the final decision epoch. Just prior to the $q$th truck move, assume:

1. $n_q \in N$ is the position of the truck,

2. $a_q(i,j)$ is the amount of goods at $i \in O$ that is destined for $j \in D$, where $a_q = \{a_q(i,j) : (i,j) \in O \times D\}$,

3. $b_q(j)$ is the amount of goods on the truck that is destined for $j \in D$, where $b_q = \{b_q(j) : j \in D\}$.

The start node, $n_1$, and the goal node, $n_T$, are both equal to the *Depot*.

Assume the vehicle arrives at node $n_q$ with current inventory position $(a_q, b_q)$. Then, if $n_q \in D$, we assume that the departing inventory position becomes $(a_q, f_1(n_q, b_q))$, where

$$[f_1(n_q, b_q)](j) = \begin{cases} b_q(j) & \text{if } j \neq n_q \\ 0 & \text{if } j = n_q \end{cases}.$$

Thus, when the vehicle arrives at a destination, it unloads all of its inventory destined for that node. Then, the decision maker selects a node in $SCS(n_q)$ to travel to next.

If $n_q \in O$, then $x(n_q, j)$ is removed from $a_q(n_q, j)$, for all $j \in D$, and loaded onto the vehicle, where in order to insure feasibility,

i. $x(i,j) \leq a(i,j) \; \forall \; i,j$

ii. $\sum_{i \in O, j \in D} x(i,j) \leq 1 - \sum_{j \in D} b(j)$

iii. $x(i,j) \geq \begin{cases} 0 & \text{if no unloading allowed at origins} \\ \sum_{j \in D} -b(j) & \text{if unloading allowed at origins} \end{cases}$

Also, the next node to visit in $SCS(n_q)$ is then selected. We remark that if $x(n_q, j)$ is negative, then a portion of the freight on the vehicle is deposited at node $n_q$. Thus, if $(a_q, b_q)$ is the inventory position when the vehicle arrives at node $n_q \in O$ and $x(n_q, j)$, for all $j \in D$, is selected, then the departing inventory position becomes $(f_2(n_q, a_q, x), f_3(n_q, b_q, x))$, where:

$$[f_2(n_q, a_q, x)](i,j) = \begin{cases} a_q(i,j) - x(i,j) & \text{if } i = n_q \\ a_q(i,j) & \text{if } i \neq n_q \end{cases}$$

$$[f_3(n_q, b_q, x)](j) = \begin{cases} b_q(j) + x(i,j) & \text{if } i = n_q \\ b_q(j) & \text{if } i \neq n_q \end{cases}$$

Therefore, for $n_q \in D$, the action set is $SCS(n_q)$, and for $n_q \in O$, the action set is $SCS(n_q) \times x(n_q, j)$ for all $j$. Let the action set $A(n, a, b, t)$ represent the set of all actions available to the driver at node $n$ at time $t$, given the state of loads to be delivered is $a$ and the loads on the vehicle is $b$.

A decision rule at time $t$ is a function $\delta(\cdot, \cdot, \cdot, t) : N \times a \times b \times [0, 1, ..., T] \rightarrow A(\cdot, \cdot, \cdot, t)$ that selects an available action. Thus, $\delta(n, a, b, t) \in A(n, a, b, t)$. A policy is a sequence of decision rules $\pi = \{\delta(\cdot, \cdot, \cdot, 1), \delta(\cdot, \cdot, \cdot, 2), ..., \delta(\cdot, \cdot, \cdot, T-1)\}$ that selects actions based on the current node and the current inventory position. We remark that $\delta(\cdot, \cdot, \cdot, t)$ is implemented only if $t$ is a decision epoch.

The cost structure is comprised of the sum of the travel costs. Let $c(n, n')$ be the cost of moving the vehicle from node $n \in N$ to node $n' \in SCS(n)$, irrespective of the amount of goods on the vehicle at the time of the move. Then, the problem criterion is

$$v^{\pi}(Depot, a, b, 1) = \sum_{q=1}^{Q} c(n_q, n_{q+1}) \tag{1}$$

conditioned on the use of policy $\pi, n = Depot, a_1 = a, b_1 = b$ and $t = 1$. The problem objective is to find a policy $\pi^*$, called an optimal policy, such that $v^{\pi^*}(Depot, a, b, 1) \leq v^{\pi}(Depot, a, b, 1)$ for all policies $\pi$ and all inventory positions $(a, b)$.

### 2.1.1 Optimality Equations

Let $v(n, a, b, t)$ be the optimal cost to be accrued from epoch $t$ through the remainder of the problem horizon, given that $n$ is the node visited at time $t$ and $(a, b)$ is the inventory position just before arrival. Then, based on the results in Puterman [79] and elsewhere, the optimality equations are as follows for $t < T$. Assuming $n \in D$,

$$v(n, a, b, t) = \min_{n'}\{c(n, n') + v(n', a, f_1(n, b), t')\}, \tag{2}$$

where the minimization is with respect to all $n' \in SCS(n)$.

If $n \in O$, then,

$$v(n, a, b, t) = \min_{n', x}\{c(n, n') + v(n', f_2(n, a, x), f_3(n, b, x), t')\}, \tag{3}$$

where the minimization is with respect to all $n' \in SCS(n)$ and all $x = \{x(n, j), j \in D\}$ such that

$$-b(n, j) \leq x(n, j) \leq a(n, j) \quad \forall \quad j \in D$$

$$\sum_j x(n, j) \leq 1 - \sum_{i,j} b(i, j).$$

If $n = Depot$, then

$$v(n, a, b, t) = \min_{n'}\{c(Depot, n') + v(n', a, b, t')\}, \tag{4}$$

where the minimum is over all $n' \in SCS(Depot)$.

The value of the solution of the optimality equation is unique, and $v(n, a, b, t) = \min_\pi v^\pi(n, a, b, t)$, for all $n, a, b$ and $t$. We refer to $v(n, a, b, t)$ as the expected cost or cost-to-go function. A necessary and sufficient condition for $\pi^*$ to be optimal is that for every $t \in \{1, ..., T\}, (a, b) \in \mathcal{F}$, and $n \in N$, the action selected causes the minimum in the optimality equations to hold.

We now show that the optimal cost increases monotonically as the number of loads to deliver, $a$, and loads on the vehicle, $b$, increases.

## 2.2 Monotonicity

Assume $(a,b),(a',b') \in \mathcal{F}$. We say $(a,b) \leq (a',b')$ if and only if $a(i,j) \leq a'(i,j)$ for all $i,j$ and $b(j) \leq b'(j)$ for all $j$. We now present conditions that imply $v(n,a,b,t)$ is monotone in $(a,b)$.

**Theorem 1** *Assume $(a,b),(a',b') \in \mathcal{F}$ and $(a,b) \leq (a',b')$. For $n \in O$, let $x^* = \{x^*(n,j)\ \forall\ j\}$, $x^* \in A(n,a',b',t)$ and $n^* \in SCS(n)$ be such that*

$$\begin{aligned}&\min\{c(n,n') + v(n', f_2(n,a',x), f_3(n,b',x), t')\}\\ &= c(n,n^*) + v(n^*, f_2(n,a',x^*), f_3(n,b',x^*), t'),\end{aligned}$$

*where the minimization is with respect to all $n' \in SCS(n)$ and all $x = \{x(n,j)\ \forall\ j\}$, where $x \in A(n,a',b',t)$. Further, assume there exists an $\tilde{x} = \{\tilde{x}(n,j)\ \forall\ j\}$, where $\tilde{x} \in A(n,a,b,t)$, that satisfies*

$$(f_2(n,a,\tilde{x}), f_3(n,b,\tilde{x})) \leq (f_2(n,a',x^*), f_3(n,b',x^*)).$$

*Then, $v(n,a,b,t) \leq v(n,a',b',t)$ for all $n$ and $t$.*

Proof: Because all loads must be serviced by the end of the time horizon, $T$, we find that $v(n,a,b,T) = v(n,a',b',T)$ for all $n$. Assume $(a,b) \leq (a',b')$ implies $v(n,a,b,t') \leq v(n,a',b',t')$ for all $n$ for $t' < T$. Then, for $n \in O$,

$$\begin{aligned}v(n,a',b',t) &= \min\{c(n,n') + v[n', f_2(n,a',x), f_3(n,b',x), t']\}\\ &= c(n,n^*) + v[n^*, f_2(n,a',x^*), f_3(n,b',x^*), t']\\ &\geq c(n,n^*) + v[n^*, f_2(n,a,\tilde{x}), f_3(n,b,\tilde{x}), t']\\ &\geq \min\{c(n,n') + v[n', f_2(n,a,x), f_3(n,b,x), t']\} = v(n,a,b,t).\end{aligned}$$

Now, consider the $n \in D$ case. It is straightforward to show that $(a,b) \leq (a',b')$ implies $(a, f_1(n,b)) \leq (a', f_1(n,b'))$. It then follows directly that $v(n,a,b,t) \leq v(n,a',b',t)$.

Finally, for the $n \in$ *Depot* case, neither $(a,b)$ nor $(a',b')$ can change from $t$ to $t'$. Therefore, $v(Depot,a,b,t) \leq v(Depot,a',b',t)$. □

Conditions are provided for the existence of $\tilde{x}$ for when a load can be removed from the vehicle at an origin. Note that if $f_2(n, a, \tilde{x}) \leq f_2(n, a', x^*)$, then $a - \tilde{x} \leq a' - x^*$.

**Corollary 2** *Let $x^* \in A(n, a', b', t)$ satisfy the conditions of the proposition. Define $\tilde{x} = \{\tilde{x}(n, j), \forall\, j\}$ as $\tilde{x}(n, j) = \max\{-b(j), x^*(n, j) + a(n, j) - a'(n, j)\}$ for all $j$. Assume $(a, b), (a', b') \in \mathcal{F}$ and $(a, b) \leq (a', b')$ as defined. Then,*

*i.* $\tilde{x} \in A(n, a, b, t)$

*ii.* $(a - \tilde{x}, b + \tilde{x}) \leq (a' - x^*, b' + x^*)$.

Proof:

(i.) By definition, if $x^* \in A(n, a', b', t)$, then $a'(n, j) \geq x^*(n, j)$ for all $j$. Therefore, if $\tilde{x}(n, j) = x^*(n, j) + a(n, j) - a'(n, j)$,

$$\begin{aligned} a'(n, j) &\geq \tilde{x}(n, j) + a'(n, j) - a(n, j) \\ &\Rightarrow a(n, j) \geq \tilde{x}(n, j). \end{aligned}$$

If $\tilde{x}(n, j) = -b(j)$, then $\tilde{x}(n, j) \leq a(n, j)$ by definition of $b(j)$.

Now, define

$$\begin{aligned} j \in D_1 &\Leftrightarrow -b(j) \geq x^*(n, j) + a(n, j) - a'(n, j) \\ j \in D_2 &\Leftrightarrow -b(j) \leq x^*(n, j) + a(n, j) - a'(n, j). \end{aligned}$$

Assume that $-b(j) \leq x^*(n, j)$ for all $n, j$ and

$$\sum_j x^*(n, j) = \sum_{j \in D_1} x^*(n, j) + \sum_{j \in D_2} x^*(n, j).$$

By definition,

$$\sum_j x^*(n, j) \leq 1 - \sum_j b'(j)$$

and

$$\sum_{j \in D_1} x^*(n, j) = \sum_{j \in D_1} -b(j),$$

such that

$$\sum_{j\in D_2} x^*(n,j) \leq 1 - \sum_{j} b'(j) + \sum_{j\in D_1} b(j).$$

Then,

$$\begin{aligned}
\sum_{j} \tilde{x}(n,j) &= \sum_{j\in D_1} \tilde{x}(n,j) + \sum_{j\in D_2} \tilde{x}(n,j) \\
&= \sum_{j\in D_1} -b(j) + \sum_{j\in D_2} (x^*(n,j) + a(n,j) - a'(n,j)) \\
&\leq \sum_{j\in D_1} -b(j) + 1 - \sum_{j} b'(j) + \sum_{j\in D_1} b(j) \\
&\quad + \sum_{j\in D_2} (a(n,j) - a'(n,j)) \\
&\leq 1 - \sum_{j} b'(j) \\
&\leq 1 - \sum_{j} b(j).
\end{aligned}$$

The first inequality is based on the assumption above, the second inequality is from

$$\sum_{j\in D_1} b(j) + \sum_{j\in D_2} a(n,j) \leq \sum_{j\in D_1} b'(j) + \sum_{j\in D_2} a'(n,j)$$

and the last inequality is true by definition of $b(j)$. Thus, $\tilde{x} \in A(n,a,b,t)$.

(ii.) By definition of $\tilde{x}$, $a(n,j) - \tilde{x}(n,j) \leq a'(n,j) - x^*(n,j)$ for all $n,j$.

Also, as $b(j) \leq b'(j)$ and $\tilde{x}(n,j) \leq x^*(n,j)$ for all $n,j$,

$$b(j) + \tilde{x}(n,j) \leq b'(j) + x^*(n,j) \ \forall\ n,j.$$

Thus, $(a - \tilde{x}, b + \tilde{x}) \leq (a' - x^*, b' + x^*)$. □

A similar set of conditions exist if the action space is restricted such that loads can not be removed from the vehicle at an origin.

**Corollary 3** *Let $x^* \in A(n,a',b',t)$ satisfy the conditions of the proposition. Define $\tilde{x} = \{\tilde{x}(n,j), \forall\, j\}$ as $\tilde{x}(n,j) = \max\{0, x^*(n,j) + a(n,j) - a'(n,j)\}$ for all $j$. Assume $(a,b),(a',b') \in \mathcal{F}$ and $(a,b) \leq (a',b')$ as defined. Then,*

*i.* $\tilde{x} \in A(n,a,b,t)$

*ii.* $(a - \tilde{x}, b + \tilde{x}) \leq (a' - x^*, b' + x^*)$.

Proof:

(i.) By definition, if $x^* \in A(n, a', b', t)$, then $a'(n,j) \geq x^*(n,j)$ for all $j$. Therefore, if $\tilde{x}(n,j) = x^*(n,j) + a(n,j) - a'(n,j)$,

$$\begin{aligned} a'(n,j) &\geq \tilde{x}(n,j) + a'(n,j) - a(n,j) \\ &\Rightarrow a(n,j) \geq \tilde{x}(n,j). \end{aligned}$$

If $\tilde{x}(n,j) = -b(j)$, then $\tilde{x}(n,j) \leq a(n,j)$ by definition of $b(j)$.

Now, define

$$\begin{aligned} j \in D_1 &\Leftrightarrow 0 \geq x^*(n,j) + a(n,j) - a'(n,j) \\ j \in D_2 &\Leftrightarrow 0 \leq x^*(n,j) + a(n,j) - a'(n,j). \end{aligned}$$

Assume that $0 \leq x^*(n,j)$ for all $n, j$ and

$$\sum_j x^*(n,j) = \sum_{j \in D_1} x^*(n,j) + \sum_{j \in D_2} x^*(n,j).$$

By definition,

$$\sum_j x^*(n,j) \leq 1 - \sum_j b'(j)$$

and

$$\sum_{j \in D_1} x^*(n,j) = 0,$$

such that

$$\sum_{j \in D_2} x^*(n,j) \leq 1 - \sum_j b'(j).$$

Then,

$$\begin{aligned} \sum_j \tilde{x}(n,j) &= \sum_{j \in D_1} \tilde{x}(n,j) + \sum_{j \in D_2} \tilde{x}(n,j) \\ &= \sum_{j \in D_2} (x^*(n,j) + a(n,j) - a'(n,j)) \\ &\leq 1 - \sum_j b'(j) + \sum_{j \in D_2} (a(n,j) - a'(n,j)) \\ &\leq 1 - \sum_j b'(j) \\ &\leq 1 - \sum_j b(j). \end{aligned}$$

The first inequality is based on the assumption above, the second inequality is from

$$\sum_{j \in D_1} b(j) + \sum_{j \in D_2} a(n,j) \leq \sum_{j \in D_1} b'(j) + \sum_{j \in D_2} a'(n,j)$$

and the last inequality is true by definition of $b(j)$. Thus, $\tilde{x} \in A(n,a,b,t)$.

(ii.) By definition of $\tilde{x}$, $a(n,j) - \tilde{x}(n,j) \leq a'(n,j) - x^*(n,j)$ for all $n, j$.

Also, as $b(j) \leq b'(j)$ and $\tilde{x}(n,j) \leq x^*(n,j)$ for all $n, j$,

$$b(j) + \tilde{x}(n,j) \leq b'(j) + x^*(n,j) \; \forall \; n, j.$$

Thus, $(a - \tilde{x}, b + \tilde{x}) \leq (a' - x^*, b' + x^*)$. □

We now present an upper bound on the benefit of the use of split loads.

## *2.3 Upper Bound on Cost Benefit*

Given that the PDPSL is a less constrained version of the PDP without the restriction that the loads have to be fully serviced on a visit, we can make the following observation.

**Observation 4** *Let $v_{PDPSL}$ be the optimal solution to the PDPSL and $v_{PDP}$ be the optimal solution to the PDP. Then,*

$$v_{PDPSL} \leq v_{PDP}.$$

Therefore, the cost of the PDPSL will always be less than or equal to that of the PDP. We now determine the upper bound on the ratio $\frac{v_{PDP}}{v_{PDPSL}}$.

The primary benefit of splitting loads is the possibility of using excess capacity on a vehicle that could not be filled if split loads are not allowed. If loads are split, there is more of an opportunity to place multiple loads on a vehicle at the same time. A cost saving may result by placing more than one load on a vehicle as it may eliminate multiple trips to serve each load individually. We define a trip as the delivery of a load or loads from a common origin to a common destination and the move to the next origin, if necessary. The more loads that can be combined on the vehicle, the higher the cost savings. We believe that the following is an upper bound on the benefit of allowing split loads.

**Conjecture 5**

$$\frac{v_{PDP}}{v_{PDPSL}} \leq 2,$$

*and the bound is tight.*

The following procedure will be used in constructing a proof of this conjecture. We consider an optimal solution to the PDPSL and eliminate split loads, converting the solution into one in which each origin/destination pair is visited a minimum number of times. We will show that the value of the constructed solution is less than or equal to $2v_{PDPSL}$.

As previously defined, a load is split if the number of pickups used to service the load is greater than the upper integer of the total load size divided by vehicle capacity. For example, if a load is 3.6 truckloads and vehicle capacity is 1, at least four trips will be required to fully service the load. If the load is picked up in five or more increments, than it is considered to be split.

We convert the PDPSL solution modifying the customers with split deliveries one at a time. If a load is split, every occurrence of that load is removed from the route. If an unsplit load is not also picked up at the origin of a split load, the location is deleted from the route and the predecessor is connected with the successor. This is also done for all destinations of split loads. These loads may now be delivered without splitting. Each load is delivered directly from the origin to the destination, with loads greater than vehicle capacity requiring the minimum number of multiple trips. These dedicated trips are placed on the route either in between loaded segments or at the end of the route, using a shortest path method that requires the destination of one of these loads to be visited directly after the origin. We now show that the cost of this converted PDP without split loads is less than or equal to $2v_{PDPSL}$.

Initially assume that all load sizes are less than or equal to vehicle capacity. This assumption will later be removed. There are several possible conditions that may be present in the PDPSL regarding the loads and how they are split:

**1)** There is never more than one split load on the vehicle at any time.

In this situation, there must be at least one load that is not split on the route. Otherwise, rerouting to eliminate split loads would result in a reduced cost. By the triangular inequality, removing any split load pickups or deliveries from a route will result in a cost that is less than or equal to the original cost. Adding the stops required to service the loads that were

previously split will also result in a cost that is less than or equal to the original cost.

Figure 2 presents a situation in which this is the case. The solution to the PDPSL results in the pickup of all of load A, followed by splitting loads B and C to deliver them in two moves each. The split loads are eliminated by removing all visits to origins and destinations associated with loads B and C from the route. This leaves a path from origin A to destination A. The full delivery of loads B and C follow this in the optimal order as dictated by the shortest path method. The cost of the original PDPSL is:

$$c(O_A, O_B) + 2c(O_B, D_B) + c(D_B, O_C) + 2c(O_C, D_C) + c(D_C, D_A)$$

while the cost of the converted route is:

$$\begin{aligned} c(O_A, D_A) + \min\{&c(D_A, O_B) + c(O_B, D_B) + c(D_B, O_C) + c(O_C, D_C), \\ &c(D_A, O_C) + c(O_C, D_C) + c(D_C, O_B) + c(O_B, D_B)\} \end{aligned}$$

By the triangular inequality, the first element $c(O_A, D_A)$ is guaranteed to be less than the PDPSL solution, such that $c(O_A, D_A) \leq v_{PDPSL}$. Similarly, the delivery of loads B and C can be shown to be less than the PDPSL solution, such that

$$\begin{aligned} &\min\{c(D_A, O_B) + c(O_B, D_B) + c(D_B, O_C) + c(O_C, D_C), \\ &c(D_A, O_C) + c(O_C, D_C) + c(D_C, O_B) + c(O_B, D_B)\} \leq v_{PDPSL} \end{aligned}$$

Therefore, the total cost of the PDP heuristic solution, $v_{HPDP}$, is less than or equal to $2v_{PDPSL}$. Because the actual solution to the PDP is less than the heuristic solution, we find the following result:

$$v_{PDP} \leq v_{HPDP} \leq 2v_{PDPSL}$$

This example also leads to the proof that the bound is tight. As the number of loads that are split along the line between the origin and destination of load A increases, the ratio $\frac{v_{PDP}}{v_{PDPSL}}$ approaches 2.

We assume that any split loads are only split once. This will hold true for loads that are split more than two times, as eliminating these additional splits does not result in any

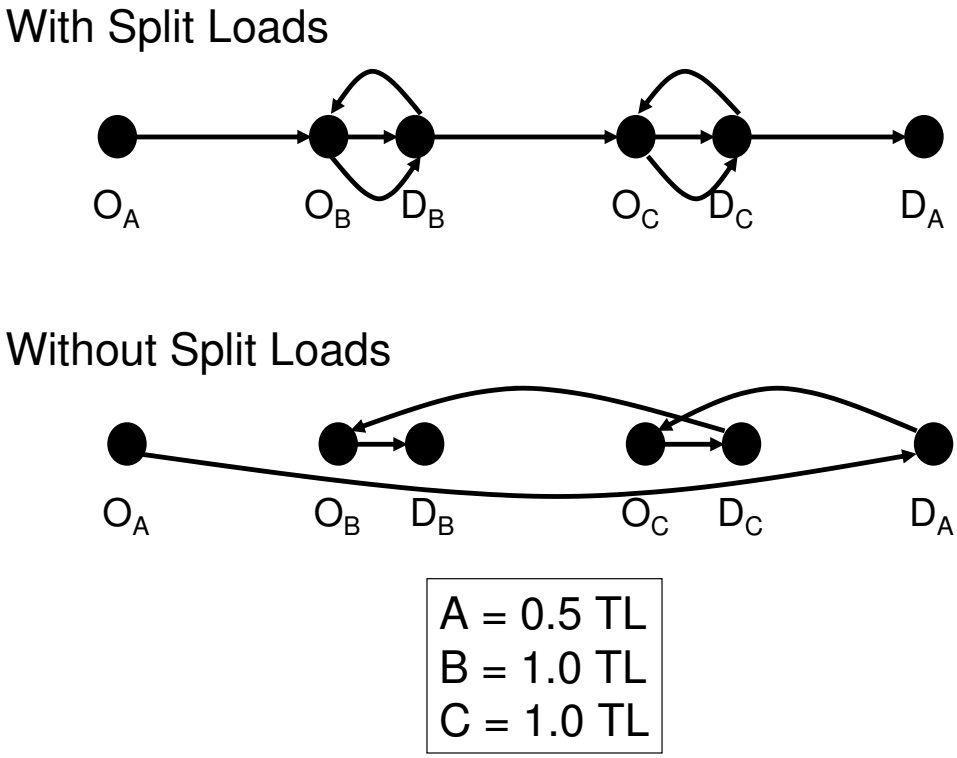


**Figure 2:** Example of situation in which cost of converted is less than or equal to two times cost of PDPSL route.

change to replacement pickups or deliveries; rather it could only reduce the overall cost by eliminating stops from the PDPSL solution. Similarly, this holds true when load sizes are greater than vehicle capacity. In this situation, not all portions of a split load must be removed from the original PDPSL. For example, if a load that is 1.5 TL must be moved, it would have to be broken into three or more splits in order to be considered a split load. If it is split into three parts, one part may remain as originally routed, while the other two are combined into one load. As shown above, this combination can be done with any two loads, so selecting the two loads may be arbitrary (without violating vehicle capacity constraints).
**2)** There is more than one split load on the vehicle at any time.

This proof will build on the proof for part 1. However, the various combinations of load sizes will be accounted for. Because of vehicle capacity, if two loads are on the vehicle at the same time, they are limited in size. This forces a limit to the size of loads that are combined together to eliminate split loads.

Archetti et al. [6] found the same bound to hold for the Split Delivery Vehicle Routing Problem.

## *2.4 Nonlinear Programming Formulation*

The dynamic programming formulation provides significant insight into the PDPSL, displaying some important structural results. However, as will be shown in Chapter 6, certain simplifications must be made in order to use dynamic programming to solve even small instances of this problem to optimality. In this section we present an alternate exact formulation of the PDPSL, as a nonlinear program. The program described here uses some of the techniques found in the formulation of the PDPSL with "Soft" Time Windows developed by Sexton and Choi [88]. We initially define the set of variables that will be used in this formulation, followed by the actual program and the modifications that were made to convert it to linear form.

### 2.4.1 Notation

The following notation was used for the nonlinear programming formulation:

Variables

$x_{in_ik}$ is the distance the vehicle has traveled on the $k$th route when visiting the $i$th origin for the $n_i$th time,

$y_{jn_jk}$ is the distance the vehicle has traveled on the $k$th route when visiting the $j$th destination for the $n_j$th time,

$z_{ijn_in_jk}$ is the amount picked up from origin $i$ on the $n_i$th visit to be dropped off at destination $j$ on the $n_j$th visit on the $k$th route,

$u_{ijn_in_jk}$ is 1 if the $n_i$th visit to origin $i$ is before the $n_j$th visit to origin $j$ on the $k$th route; 0 otherwise,

$v_{ijn_in_jk}$ is 1 if the $n_i$th visit to origin $i$ is before the $n_j$th visit to destination $j$ on the $k$th route; 0 otherwise,

$w_{ijn_in_jk}$ is 1 if the $n_i$th visit to destination $i$ is before the $n_j$th visit to destination $j$ on the $k$th route; 0 otherwise,

$r_{in_ik}$ is 1 if origin $i$ is visited $n_i$ times on route $k$; 0 otherwise,

$t_{jn_jk}$ is 1 if destination $i$ is visited $n_i$ times on route $k$; 0 otherwise,

$y_{depot,k}$ is the distance the vehicle has traveled when it returns to the depot on the $k$th route,

$p_k$ is 1 if route $k$ is used; 0 otherwise.

Parameters

$a_{ij}$ is the amount required for delivery between origin $i$ and destination $j$,

$c_{ij}$ is the distance between node $i$ and node $j$, assuming that costs are symmetric such that $c_{ij} = c_{ji}$.

Indices

$O$ is the set of origins, $D$ the set of destinations, $N_i$ the number of possible stops at origin (or destination) $i$ on a route, $K$ is the maximum number of possible routes (dependent on the total loads to be delivered) and $M$ is a large number.

### 2.4.2 Nonlinear Programming Formulation

The nonlinear programming formulation can now be stated as:

$\min \sum_{k \in K} y_{depot,k}$

subject to

$$\sum_{j \in D} \sum_{n_j \in N_j} z_{ijn_in_jk} \leq 1 - \sum_{l \in O: l \neq i || n_l < n_i} \sum_{j \in D} \sum_{n_l \in N_l} \sum_{n_j \in N_j} z_{ljn_ln_jk} u_{lin_ln_ik} v_{ijn_in_jk}$$
$$i \in O, n_i \in N_i, k \in K \tag{5}$$

$$c_{ij} r_{in_ik} r_{jn_jk} - M(1 - r_{in_ik}) \leq x_{in_ik} - x_{jn_jk} + M u_{ijn_in_jk} \leq M - c_{ij} r_{in_ik} r_{jn_jk}$$
$$i, j \in O, i \neq j, n_i, n_j \in N_j, k \in K \tag{6}$$

$$c_{ij} r_{in_ik} t_{jn_jk} - M(1 - r_{in_ik}) \leq x_{in_ik} - y_{jn_jk} + M v_{ijn_in_jk} \leq M - c_{ij} r_{in_ik} t_{jn_jk}$$
$$i \in O, j \in D, n_i, n_j \in N_j, k \in K \tag{7}$$

$$c_{ij} t_{in_ik} t_{jn_jk} - M(1 - t_{in_ik}) \leq y_{in_ik} - y_{jn_jk} + M w_{ijn_in_jk} \leq M - c_{ij} t_{in_ik} t_{jn_jk}$$
$$i, j \in D, i \neq j, n_i, n_j \in N_j, k \in K \tag{8}$$

$$y_{depot,k} - y_{jn_jk} \geq c_{j,depot} p_k \quad j \in D, n_j \in N_j, k \in K \tag{9}$$

$$x_{i1k} \geq c_{i,depot} p_k \quad i \in O, k \in K \tag{10}$$

$$r_{in_ik} \leq M \sum_{l \in D} \sum_{n_l \in N_l} z_{iln_in_lk} \quad i \in O, n_i \in N_i, k \in K \tag{11}$$

$$M r_{in_ik} \geq \sum_{l \in D} \sum_{n_l \in N_l} z_{iln_in_lk} \quad i \in O, n_i \in N_i, k \in K \tag{12}$$

$$t_{jn_jk} \leq M \sum_{l \in O} \sum_{n_l \in N_l} z_{ljn_ln_jk} \quad j \in D, n_j \in N_j, k \in K \tag{13}$$

$$M t_{jn_jk} \geq \sum_{l \in O} \sum_{n_l \in N_l} z_{ljn_ln_jk} \quad j \in D, n_j \in N_j, k \in K \tag{14}$$

$$u_{ijn_in_jk} \leq M \sum_{l \in D} \sum_{n_l \in N_l} z_{iln_in_lk} \quad i, j \in O, n_i \in N_i, n_j \in N_j, k \in K \tag{15}$$

$$u_{ijn_in_jk} \leq M \sum_{l \in D} \sum_{n_l \in N_l} z_{jln_jn_lk} \quad i, j \in O, n_i \in N_i, n_j \in N_j, k \in K \tag{16}$$

$$u_{iin_im_ik} = r_{im_ik} \quad i \in O, n_i, m_i \in N_i, m_i > n_i, k \in K \tag{17}$$

$$v_{ijn_in_jk} \leq M \sum_{l \in D} \sum_{n_l \in N_l} z_{iln_in_lk} \quad i \in O, j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{18}$$

$$v_{ijn_in_jk} \leq M \sum_{l \in O} \sum_{n_l \in N_l} z_{ljn_ln_jk} \quad i \in O, j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{19}$$

$$w_{ijn_in_jk} \leq M \sum_{l \in O} \sum_{n_l \in N_l} z_{lin_ln_ik} \quad i, j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{20}$$

$$w_{ijn_in_jk} \leq M \sum_{l \in O} \sum_{n_l \in N_l} z_{ljn_ln_jk} \quad i, j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{21}$$

$$w_{jjn_jm_jk} = t_{jm_jk} \quad j \in D, n_j, m_j \in N_j, m_j > n_j, k \in K \tag{22}$$

$$\sum_{i\in O}\sum_{j\in D}\sum_{n_i\in N_i}\sum_{n_j\in N_j}\sum_{k\in K} z_{ijn_in_jk} = a_{ij} \tag{23}$$

$$x_{in_ik} \leq Mr_{in_ik} \quad i \in O, n_i \in N_i, k \in K \tag{24}$$

$$x_{in_ik} \geq r_{in_ik} \quad i \in O, n_i \in N_i, k \in K \tag{25}$$

$$y_{jn_jk} \leq Mt_{jn_jk} \quad j \in D, n_j \in N_j, k \in K \tag{26}$$

$$y_{jn_jk} \geq t_{jn_jk} \quad j \in D, n_j \in N_j, k \in K \tag{27}$$

$$\sum_{i\in O}\sum_{n_i\in N_i} x_{in_ik} \leq Mp_k \quad k \in K \tag{28}$$

$$\sum_{j\in D}\sum_{n_j\in N_j} y_{jn_jk} \leq Mp_k \quad k \in K \tag{29}$$

$$\sum_{i\in O}\sum_{j\in D}\sum_{n_i\in N_i}\sum_{n_j\in N_j} z_{ijn_in_jk} \leq Mp_k \quad k \in K \tag{30}$$

$$x_{in_ik} + \epsilon \leq x_{in_i+1k} + M(1 - r_{in_i+1k}) \quad i \in O, n_i + 1 \in N_i, k \in K \tag{31}$$

$$y_{jn_jk} + \epsilon \leq y_{jn_j+1k} + M(1 - t_{jn_j+1k}) \quad j \in D, n_j + 1 \in N_j, k \in K \tag{32}$$

$$r_{in_ik} \geq r_{in_i+1k} \quad i \in O, n_i \in N_i, k \in K \tag{33}$$

$$t_{jn_jk} \geq t_{jn_j+1k} \quad j \in D, n_j \in N_j, k \in K \tag{34}$$

$$z_{ijn_in_jk} \leq v_{ijn_in_jk} \quad i \in O, j \in D, n_i, n_j \in N_j, k \in K \tag{35}$$

$$p_{k+1} \leq p_k \quad k \in K \tag{36}$$

$$x_{in_ik} \geq 0 \quad \forall \quad i \in O, n_i \in N_i, k \in K \tag{37}$$

$$y_{jn_jk} \geq 0 \quad \forall \quad j \in D, n_j \in N_j, k \in K \tag{38}$$

$$z_{ijn_in_jk} \geq 0 \quad \forall \quad i \in O, j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{39}$$

$$y_{depot,k} \geq 0 \quad \forall k \in K \tag{40}$$

$$u_{ijn_in_jk} \in \{0,1\} \quad \forall \quad i,j \in O, n_i \in N_i, n_j \in N_j, k \in K \tag{41}$$

$$v_{ijn_in_jk}, e_{ijn_in_jk} \in \{0,1\} \quad \forall \quad i \in O, j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{42}$$

$$w_{ijn_in_jk} \in \{0,1\} \quad \forall \quad i,j \in D, n_i \in N_i, n_j \in N_j, k \in K \tag{43}$$

$$r_{in_ik} \in \{0,1\} \quad \forall \quad i \in O, n_i \in N_i, k \in K \tag{44}$$

$$t_{jn_jk} \in \{0,1\} \quad \forall \quad j \in D, n_j \in N_j, k \in K \tag{45}$$

$$p_k \in \{0,1\} \quad \forall k \in K \tag{46}$$

The objective is to minimize the distance that the vehicle has traveled when it returns to the depot on each route. Constraint 5 keeps the loads on the vehicle within the capacity. The amount picked up at an origin must be less than or equal to the capacity minus the amount that has already been picked up, but not yet dropped off. Constraints 6-8 maintain the order of origins and destinations and force the program to correctly calculate the distances required for travel. The $r, t$ binaries indicate that the constraint is only to be used when the stops involved in the constraint are both visited the number of times indicated by the $n_i, n_j$. A node can not be visited in less than the distance required to get to the previous node plus the distance from the previous node to the current node. Constraint 9 guarantees that if a route is used, the route ends at the depot. Similarly, constraint 10 guarantees that if a route is used, the route begins at the depot. Constraints 11 and 12 indicate that an origin

is only visited if a load is picked up and vice versa. Constraints 13 and 14 do the same for destinations. Constraints 15-22 give conditions for the precedence variables, forcing them to zero if a stop is not visited. Constraint 23 forces all loads to be picked up. Constraints 24-27 prevent a stop from being visited if no distance is traveled to reach that stop and vice versa. Constraints 28 and 29 prohibit stops on a route from being visited if the route is not used. Similarly, constraint 30 does not allow loads to be delivered on a route that doesn't exist. Constraints 31-34 prevent stops from being visited out of order based on $n_i$. Constraint 35 does not allow a load to be delivered if the destination is visited before the origin. Constraint 36 maintains the order of the routes.

### 2.4.3 Linear Conversion

Given that each of the nonlinear terms can be broken down into bilinear components and at least one of the variables in the nonlinear term is binary, the formulation can easily be transformed into a mixed integer program. This conversion allows for the use of a linear programming solver such as CPLEX. An additional constraint required to allow for this transformation is that if one of the variables is not binary, it must be bound to the binary variable such that the value of the nonbinary variable is zero when the binary variable is zero. Now, given the bilinear term $xy$ in which, without loss of generality, $x$ is binary, $xy$ can be converted to:

$$\begin{aligned} z &= xy \\ z &\leq x \\ z &\leq y \\ z &\geq x + y - 1 \end{aligned}$$

Al-Khayyal [1] shows that this transformation is equivalent to the original formulation.

The following variables are introduced:

$s_{ijn_in_jk} = r_{in_ik}t_{jn_jk} = 1$ if origin $i$ is visited $n_i$ times and destination $j$ is visited $n_j$ times on route $k$; 0 otherwise.

$o_{ijn_in_jk} = r_{in_ik}r_{jn_jk} = 1$ if origin $i$ is visited $n_i$ times and origin $j$ is visited $n_j$ times on

route $k$; 0 otherwise.

$q_{ijn_in_jk} = t_{in_ik}t_{jn_jk} = 1$ if destination $i$ is visited $n_i$ times and destination $j$ is visited $n_j$ times on route $k$; 0 otherwise.

$g_{lijn_ln_in_jk} = u_{lin_ln_ik}v_{ijn_in_jk} = 1$ if the $n_l$th visit to origin $l$ is before the $n_i$th visit to origin $i$ and the $n_i$th visit to origin $i$ is before the $n_j$th visit to destination $j$ on the $k$th route; 0 otherwise.

$h_{lijn_ln_in_jk} = g_{lijn_ln_in_jk}z_{ljn_ln_jk} = z_{ljn_ln_jk}$ if $g_{lijn_ln_in_jk} = 1$; 0 otherwise.

Three constraints must be added for each new variable. Constraints 47-53 are the transformation of constraint 5. The transformation of the distance constraints are constraints 54-57 for constraint 6, constraints 58-62 for constraint 7, and constraints 63-66 for constraint 8.

$$\sum_{j\in D}\sum_{n_j\in N} z_{ijn_in_jk} \leq 1 - \sum_{l\in O: l\neq i || n_l<n_i}\sum_{j\in D}\sum_{n_l\in N}\sum_{n_j\in N} h_{lijn_ln_in_jk} \quad i\in O, n_i\in N, k\in K \tag{47}$$

$$g_{lijn_ln_in_jk} \leq u_{lin_ln_ik} \quad l,i\in O, j\in D, n_l,n_i,n_j\in N, k\in K \tag{48}$$

$$g_{lijn_ln_in_jk} \leq v_{ijn_in_jk} \quad l,i\in O, j\in D, n_l,n_i,n_j\in N, k\in K \tag{49}$$

$$g_{lijn_ln_in_jk} \geq u_{lin_ln_ik} + v_{ijn_in_jk} - 1 \quad l,i\in O, j\in D, n_l,n_i,n_j\in N, k\in K \tag{50}$$

$$h_{lijn_ln_in_jk} \leq g_{lijn_ln_in_jk} \quad l,i\in O, j\in D, n_l,n_i,n_j\in N, k\in K \tag{51}$$

$$h_{lijn_ln_in_jk} \leq z_{ljn_ln_jk} \quad l,i\in O, j\in D, n_l,n_i,n_j\in N, k\in K \tag{52}$$

$$h_{lijn_ln_in_jk} \geq g_{lijn_ln_in_jk} + z_{ljn_ln_jk} - 1 \quad l,i\in O, j\in D, n_l,n_i,n_j\in N, k\in K \tag{53}$$

$$c_{ij}o_{ijn_in_jk} - M(1-r_{in_ik}) \leq x_{in_ik} - x_{jn_jk} + Mu_{ijn_in_jk} \leq M - c_{ij}o_{ijn_in_jk}$$
$$i,j\in O, i\neq j, n_i,n_j\in N, k\in K \tag{54}$$

$$o_{ijn_in_jk} \leq r_{in_ik} \quad i,j\in O, i\neq j, n_i,n_j\in N, k\in K \tag{55}$$

$$o_{ijn_in_jk} \leq r_{jn_jk} \quad i,j \in O, i \neq j, n_i, n_j \in N, k \in K \tag{56}$$

$$o_{ijn_in_jk} \geq r_{in_ik} + r_{jn_jk} - 1 \quad i,j \in O, i \neq j, n_i, n_j \in N, k \in K \tag{57}$$

$$c_{ij}s_{ijn_in_jk} - M(1 - r_{in_ik}) \leq x_{in_ik} - y_{jn_jk} + Mv_{ijn_in_jk} \leq M - c_{ij}s_{ijn_in_jk}$$
$$i \in O, j \in D, n_i, n_j \in N, k \in K \tag{58}$$

$$s_{ijn_in_jk} \leq r_{in_ik} \quad i \in O, j \in D, n_i, n_j \in N, k \in K \tag{59}$$

$$s_{ijn_in_jk} \leq t_{jn_jk} \quad i \in O, j \in D, n_i, n_j \in N, k \in K \tag{60}$$

$$s_{ijn_in_jk} \geq r_{in_ik} + t_{jn_jk} - 1 \quad i \in O, j \in D, n_i, n_j \in N, k \in K \tag{61}$$

$$s_{ijn_in_jk} \geq v_{ijn_in_jk} \quad i \in O, j \in D, n_i, n_j \in N, k \in K \tag{62}$$

$$c_{ij}q_{ijn_in_jk} - M(1 - t_{in_ik}) \leq y_{in_ik} - y_{jn_jk} + Mw_{ijn_in_jk} \leq M - c_{ij}q_{ijn_in_jk}$$
$$i,j \in D, i \neq j, n_i, n_j \in N, k \in K \tag{63}$$

$$q_{ijn_in_jk} \leq t_{in_ik} \quad i,j \in O, i \neq j, n_i, n_j \in N, k \in K \tag{64}$$

$$q_{ijn_in_jk} \leq t_{jn_jk} \quad i,j \in O, i \neq j, n_i, n_j \in N, k \in K \tag{65}$$

$$q_{ijn_in_jk} \geq t_{in_ik} + t_{jn_jk} - 1 \quad i,j \in O, i \neq j, n_i, n_j \in N, k \in K \tag{66}$$

## *2.5 Conclusions*

In this chapter, we introduced the Pickup and Delivery Problem with Split Loads (PDPSL). The problem was formulated as a dynamic program and the cost was shown to be monotonically increasing with the size of loads to be delivered and the size of loads on the vehicle. The upper bound on the benefit of using split loads was shown to be one half of the cost of a routing solution without split loads. Finally, an alternative exact formulation was presented, consisting of a nonlinear program that was converted into a mixed integer program.

# CHAPTER III

# PDPSL HEURISTIC

The computational intractability of the PDP makes it difficult to determine an exact solution for even some of the smallest problems. Therefore, most research has focused on the development of heuristics for finding approximate solutions. In this chapter, we present a heuristic that uses elements of a "classical" construction heuristic and newer "metaheuristic" methods to solve the PDPSL. Section 3.1 provides a background on the methods applied to the heuristic described in Section 3.2. Section 3.3 expands on this description with a specific framework for the heuristic. Conclusions are presented in Section 3.4.

## *3.1 Heuristic Methods*

The complexity of the Vehicle Routing Problem (VRP), and the more general Pickup and Delivery Problem, precludes the exact solution of any realistically sized problem in a reasonable length of time. Therefore, researchers have turned to heuristics for help in approximating good solutions. Cordeau et al. [13] provide a thorough review of various VRP heuristics. VRP heuristics have steadily evolved over the last 40 years. "Classical" construction heuristics put an emphasis on quickly obtaining a feasible solution with the possibility of later using an improvement procedure. One of these heuristics is the well-known Clarke and Wright Savings Algorithm [12]. This algorithm remains widely used in practice despite some shortcomings involving the accuracy of the solutions provided and the adaptability of the heuristic to a wide range of problems. A main reason for its continued popularity is the speed with which it can obtain a "good" solution. The algorithm creates an initial feasible solution by dedicating an individual route to each customer back and forth from the depot. At each iteration, two routes are merged into a single route whenever this is feasible, thus generating a cost saving. Figure 3 presents a simple example of this heuristic. Although the heuristic does not necessarily find the optimal solution, it has been shown to generate

a good solution with negligible computing time [55].

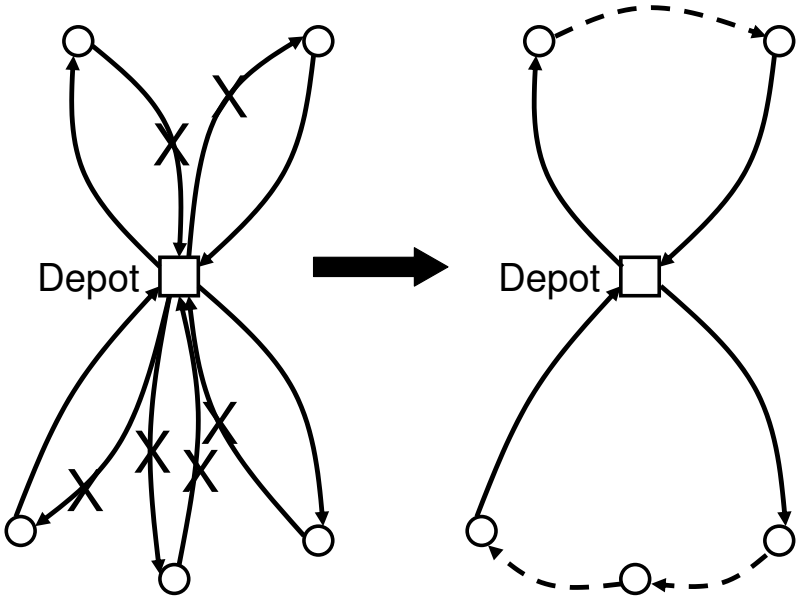


**Figure 3:** Example showing the use of the Clarke and Wright Savings Algorithm for the Vehicle Routing Problem.

Recent research has focused more heavily on improvement heuristics, using two main principles: local search and population search. Local search intensively explores the solution space by moving at each step from the current solution to another promising solution in its neighborhood. The neighborhood is defined as those solutions that can be reached from the original solution by making a slight modification to the current solution. These modifications may include the insertion of a load from one route to another, the swap of two loads, or more complicated procedures involving multiple loads and moves.

Population search utilizes a pool of good parent solutions from which it selects certain traits to create offspring solutions. Simulated annealing [52] and tabu search [37] are two heuristics that use local search, while genetic search [48] is an example of population search. Each of these methods implements some process that prevents the solution from becoming fixed at a local minimum. That is, they allow steps that are detrimental to the objective function, moving the solution away from a local minimum to test other possibilities.

The tabu search heuristic was first introduced and thoroughly explored by Glover [37, 38, 39]. Tabu search performs a local search by moving at each iteration from the current

solution to a neighboring solution. This move may be inferior, causing a depreciation of the objective function, so an anti-cycling mechanism is utilized to prevent a return to the original solution. This mechanism declares certain attributes of the original solution as tabu, or forbidden, for a number of iterations. Therefore, the algorithm forces a move away from the local minimum. Tabu search has been shown to significantly outperform most other metaheuristics [35].

## *3.2 PDPSL Heuristic*

The heuristic presented here combines the traditional features of the Clarke and Wright Savings Algorithm with some of the techniques used in tabu search, as well as some common local search techniques, for the PDPSL. An initial solution is obtained by dedicating a route to each individual load. A load is then selected to be split based on the additional cost of generating the split. A simple tabu search is applied to prevent the selection of the same split loads repeatedly. Routes are combined through a technique similar to Clarke and Wright, with combinations selected based on reduction in cost and constrained by the capacity of the vehicle at each stop along a route. Several local search techniques are utilized to improve the solution, including:

1. Intra-route load swaps
2. Inter-route load swaps
3. Intra-route load insertions
4. Inter-route load insertions
5. Reordering of origins or destinations

A tabu list of visited solutions is avoided for a number of iterations in an attempt to analyze a broader neighborhood within the solution space. This tabu list is associated with the creation of split loads. Because the act of splitting a load can only increase cost (in some instances the cost may not change), this is initially an undesirable move. That is, by splitting a load, the cost of travel between the origin and destination of the load is added

a second time. However, the split that generates the lowest cost is not generally the best split to select. Therefore, the split is randomly selected, as described below. Because a split is the first change made after initial route generation, the selection of a split guides the formation of a solution. A list is maintained to prohibit the selection of the same first split load in multiple iterations. A similar list is maintained for the second split selected. Any additional splits are further tracked using a more traditional tabu list. For example, solution $X$ is modified by splitting $1/y$ truckloads of load $L$, creating solution $X'$. If $X$ is a good solution, the algorithm may return to it. However, if we return to solution $X$, the split of $1/y$ truckloads of load $L$ is prohibited from occurring for several iterations. Similarly to traditional tabu search, this procedure allows for a search of a more diverse solution space.

A description of the procedures involved in this heuristic is now presented.

### 3.2.1 Route Generation

A route is created that runs from the depot, to the origin, to the destination, returning to the depot, for each load request that is equal to or below vehicle capacity. A load request that is greater than vehicle capacity is partitioned, with routes initially carrying full vehicle loads, until there are enough routes to service the entire load. Figure 4 presents an example of route generation. Load $A$ has 1.4 truckloads to be delivered, and the first route generated carries a full truckload, while the remaining 0.4 truckloads is delivered by the second route. Loads $B$ and $C$ are both less than or equal to vehicle capacity, so they can each be serviced by one route.

### 3.2.2 Split Load Creation

A load is selected from any route and the size of this load is compared to all instances of excess capacity on any route, at any point within the route. If the size of the load is greater than any of these occurrences of capacity, it may be split. The load is split such that the portion of the load that is moved has the same size as the excess capacity at the location in the route that it is moved to. That is, if a load of size 0.8 truckloads is split, and the excess capacity to where it is moved is 0.5 truckloads, 0.5 truckloads will be split off, while 0.3 truckloads remains in the original route. A load may be split and moved to another route,

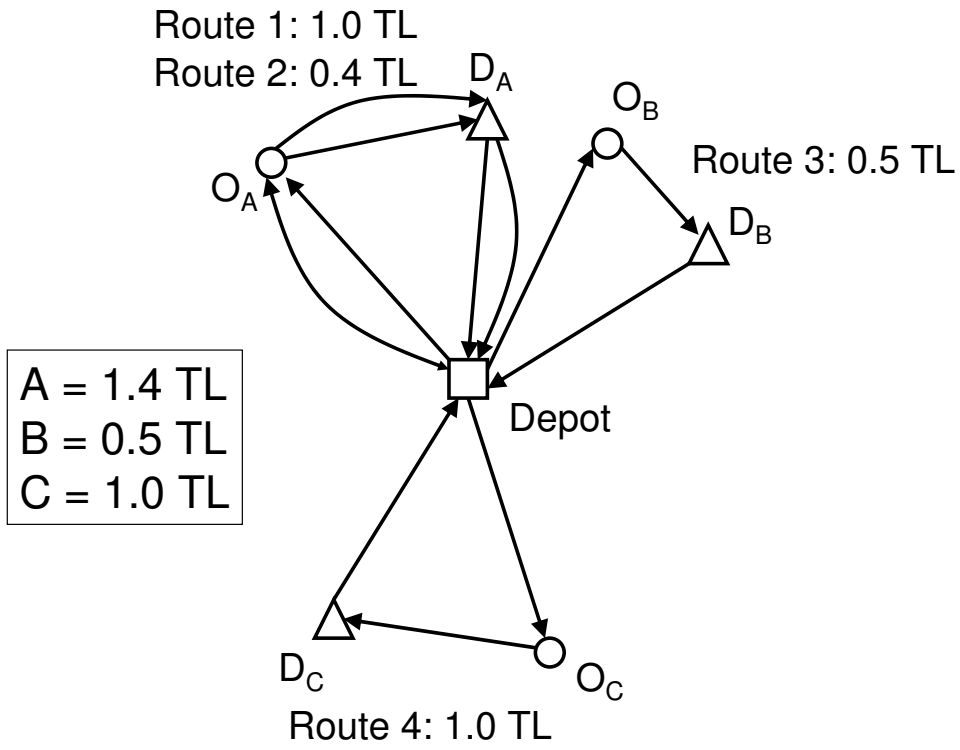


**Figure 4:** Route generation example with unit capacity vehicles.

or to another position within the same route. Based on this procedure, a split load can not be added to the end of a route as the excess capacity is the full vehicle capacity and no split can be created. Figure 5 presents an example of how a split load is created. Assume that load $B$ is to be split. The only excess capacity in this system occurs on route 1, after load $A$ has been picked up, with 0.6 truckloads of spare capacity available. Therefore, $B$ is split such that 0.6 is picked up by route 1, and 0.3 is left on route 2. In this example (not pictured), load $A$ could also be split such that 0.1 truckloads is transported on route 2 and the remaining 0.3 is left on route 1.

The split is checked against the tabu lists to confirm that it has not been recently considered. A split load may also be rejected if it is a load that was earlier inserted as a local improvement. This prevents cycling with other parts of the heuristic. It is also checked to ensure that it has not been previously selected and rejected. A route modification may be rejected for the following two reasons:

1. the number of split loads on any route is greater than a fixed parameter

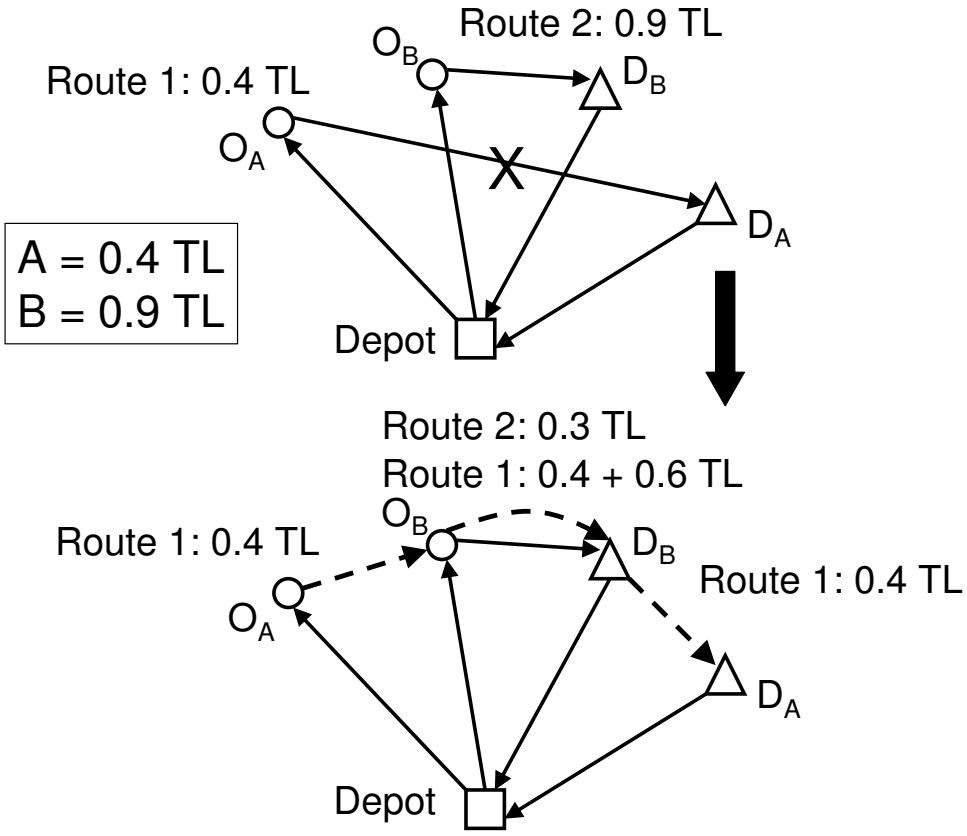


**Figure 5:** Split load creation example in which load $B$ is split and placed on route 1.

2. the new route violates the maximum route length.

The parameters associated with both of these constraints may be adjusted based on the situation. Each section of the heuristic that modifies a route tests for these violations. These constraints cannot be violated under the conditions of the problem discussed here. However, they will be applied in later tests and we discuss them in more detail then. The computational time required for these tests is negligible, so including them now does not affect the results.

As the algorithm continues, if the split is not on any tabu list, the cost of the modification is determined, and a random number is generated dependent on this cost. In the example in Figure 5, the cost is:

$$c(O_A, O_B) + c(O_B, D_B) + c(D_B, D_A) - c(O_A, D_A)$$

By the triangular inequality, it can be seen that this cost must be greater than or equal to zero. The ratio of this cost to the overall route cost is determined, and a random number is selected based on this ratio. This random number determines whether or not the split is selected. Any split, regardless of cost, has some likelihood of selection. However, a split load that generates a smaller cost has a greater probability of selection.

After a load is selected, the new routes are tested against the rules described above applying to route length and number of split loads. If neither of these restrictions are violated, the split is accepted and added to the list *tabusplit*. If a violation occurs, the split is rejected and another split is tested. This is repeated for a finite number of iterations. If a valid split can not be found, the algorithm moves on to make local improvements to the routes, and then repeats the search for a feasible split load. If all possible split loads are not accepted based on cost, then the algorithm also continues, without repeating iterations to search for more split loads.

### 3.2.3 Route Combination

In this section, two routes are selected and the cost of combining them is initially determined. The routes may be combined such that one is placed after another, which most closely resembles the Clarke and Wright technique used for the VRP. If capacity constraints allow it, one route may also be inserted after an origin or destination on another route. Figure 6 presents two possible route combinations. Route 1 and Route 2 are combined such that Route 2 is simply added after the end of Route 1. No other combination of these two routes is possible, as the vehicle capacity constraint would be violated. Capacity constraints do allow for Route 4 to be inserted into Route 3, after the origin of load $C$. There is 0.5 truckloads of excess capacity after picking up load $C$, which is enough to fit load $D$.

Each route combination is initially checked to determine whether it has not already been tested and rejected for violating one of the two conditions discussed in Section 3.2.2. Next, the length of the new route is evaluated to determine if it will violate the maximum length. Unlike the split load creation, the routes are initially tested against the maximum allowable length plus some buffer. The buffer is a small value relative to route length, selected to allow

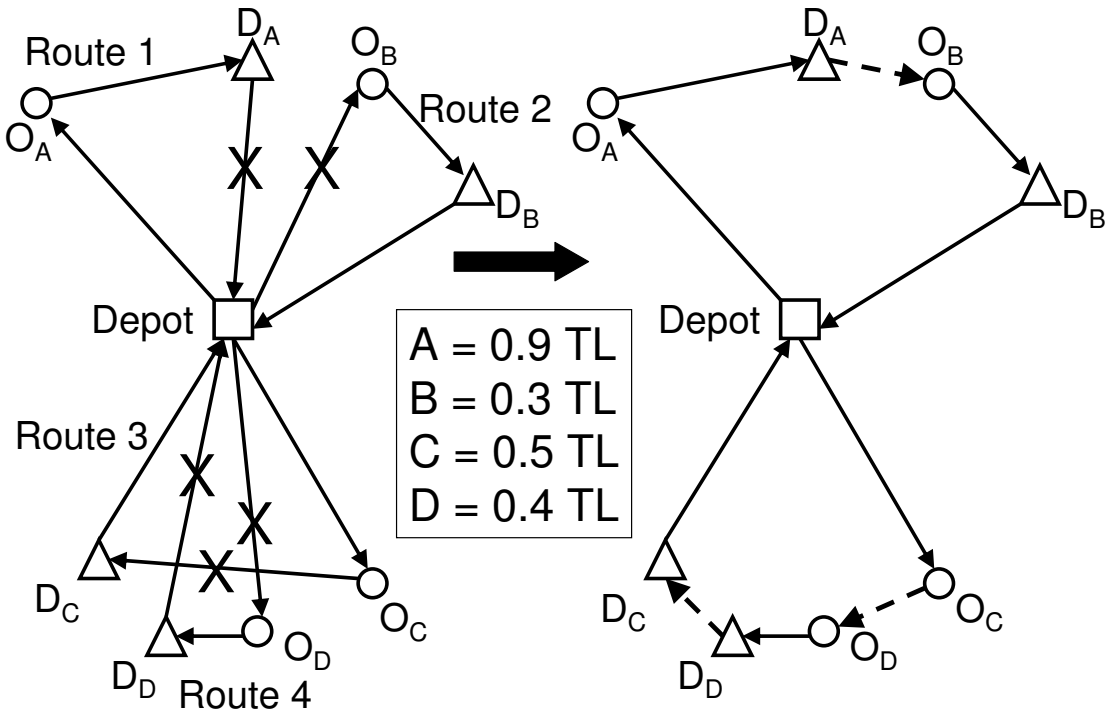


**Figure 6:** Example of possible cost saving route combinations.

for the creation of a route that initially violates the constraint. An infeasible route may be selected because a local improvement described in the following paragraph is performed on the route that may reduce the length. The route is tested against the maximum allowable length without the buffer after the local improvement is performed. Before the improvement, if the length is greater than the maximum allowable length plus the buffer, the combination is rejected.

The cost of each feasible route combination is then determined. If there is at least one combination that reduces the overall cost, the combination that has the maximum reduction in cost is selected. The routes are tested to determine if the combination creates a violation of the restriction on the number of split loads per route. If not, a local improvement is performed by switching origins or destinations within a route. If two origins are visited consecutively on a route, their order may be switched if a cost improvement results (similarly with two destinations). Because only consecutive origins (or destinations) are swapped, this is an effortless move that has no effect on vehicle capacity constraints, or on the requirement

that the origin of a load be visited prior to the destination. This procedure is performed after each combination, swap or insertion. This process is repeated until no cost improvement is found through the combination of routes.

### 3.2.4 Intra-Route Load Insertion

This section analyzes each load to be potentially inserted from one loaded segment of a route to another. A *loaded segment* of a route is a section on which the vehicle begins and ends with full capacity available. These segments are similar to the "mini-clusters" used by Ioachim [49]. The beginning of a loaded segment is when the vehicle is either beginning the route or is at an origin completely empty after having dropped off all remaining loads at the prior destination. A loaded segment must end at a destination, at which it delivers all remaining loads. Because of these guidelines, loaded segments within a route may be treated as separate routes in terms of capacity constraints. That is, any changes to the loads delivered on a loaded segment only have an effect on that segment, and not on the rest of the route. Also, this subalgorithm, as well as the intra-route load swap, are only performed if there are two or more loaded segments on a route.

Insertion involves moving the entire load to another part of the route. The load insertion is evaluated to determine if capacity constraints are violated, if the insertion has already been attempted, or if the insertion would return a split load back to its original position. Splitting a load generates an additional cost; therefore, inserting the split portion back in its original location is a cost saving move. This must be prevented in order to allow for split loads to be utilized. A load may be inserted anywhere within the loaded segment: after an origin, destination, or at the end of the route (if the loaded segment is the last one on the route). The load is inserted such that its origin and destination are placed consecutively after the stop selected. If more than one load was picked up from the origin in its original location on the route, it is left on the route; otherwise, it is eliminated. This applies to the destination as well. Figure 7 presents an example of a possible load insertion. Loads $A, B$ and $D$ are initially delivered on loaded segment 1, while load $C$ is on loaded segment 2. The load size of $D$ allows for it to be inserted after load $C$ has been picked up on segment 2. No

other loads are picked up or delivered to the origin or destination for load $D$ on segment 1, so they are eliminated from that portion of the route.

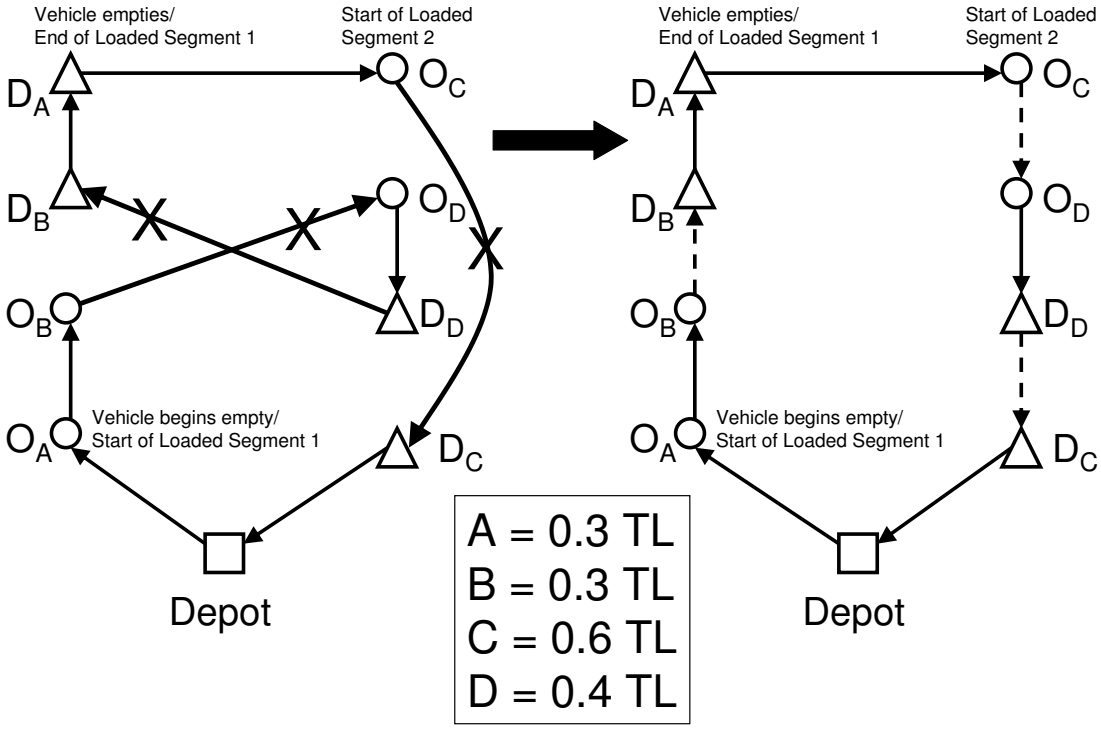


**Figure 7:** Intra-route load insertion example with load $D$ inserted into segment 2.

The cost of each feasible insertion is determined. If at least one insertion reduces the overall cost, the insertion that results in the maximum reduction in cost is selected. The routes are tested to determine if the insertion creates a violation of the restriction on the number of split loads per route. The local improvement of switching origins or destinations within the route is again performed. If the length of the modified route is less than the maximum route length, the insertion is accepted. Loads are further inserted within each route until no cost improvement is found.

### 3.2.5 Intra-Route Load Swapping

In this exchange procedure, two loads from different loaded segments of a route are selected to be potentially swapped. In a swap of two loads, $A$ and $B$, the origin for load $B$ is moved to the order in the route previously held by the origin for load $A$, and vice versa. The

same is true for the destinations. Because of this, there may be multiple stops between the origin and destination for a load that has been swapped, or the two stops may be visited consecutively on the route. If other loads are picked up at the origin for a load that is swapped, the origin is still visited, and the new origin that is swapped in is visited immediately afterwards. If no other loads are picked up at the origin, it is eliminated, and the new origin fills the slot. This also applies to the destinations. Figure 8 presents a potential load swap. Loads $A$ and $D$ are initially delivered on loaded segment 1, while loads $B$ and $C$ are on loaded segment 2. The load sizes of $B$ and $D$ are such that they may be swapped without violating the vehicle capacity constraint. Neither the origins nor the destinations for loads $B$ and $D$ apply to other loads, so they may be eliminated from their original segments.

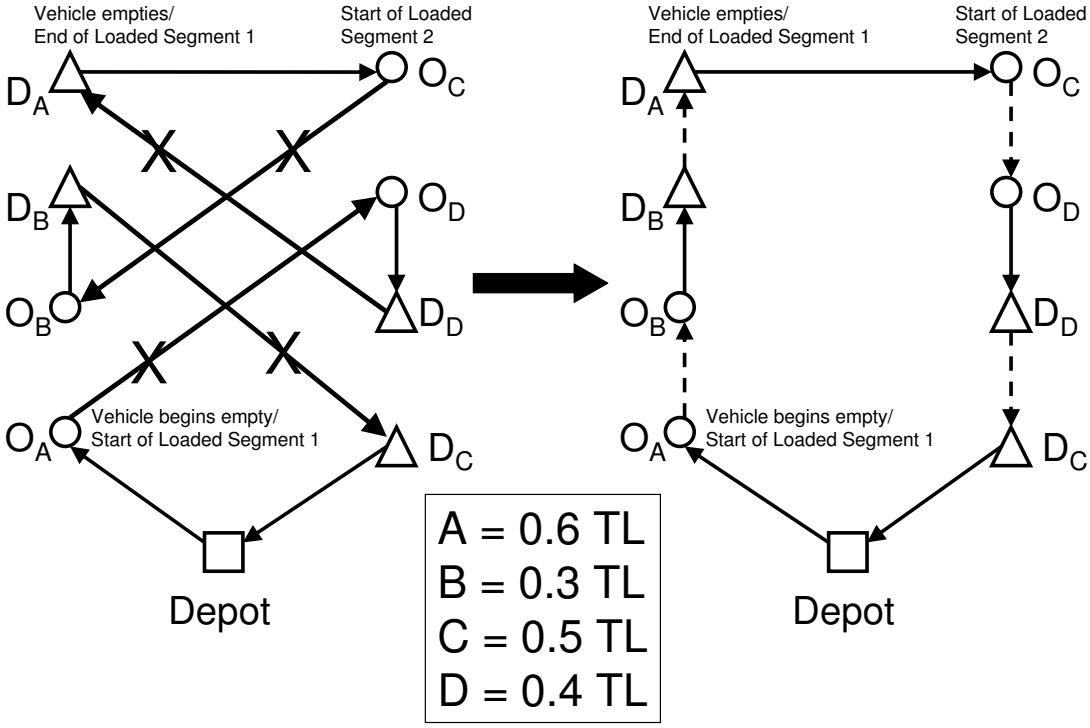


**Figure 8:** Intra-route load swap example with loads $B$ and $D$ swapped for each other.

The swap of the two loads is evaluated to determine if capacity constraints are violated, or if the swap has already been attempted. A swap may have been previously rejected for a violation of one of the two constraints. However, if it was recently accepted, it is also

rejected. This is to prevent cycling with the insertion heuristic. A swap may be effectively reversed through a few insertion moves. Therefore, it is not repeated for several iterations.

The cost of each feasible swap is determined. If at least one swap reduces the overall cost, the swap of the two loads with the maximum reduction in cost is selected. The routes are again tested to determine if the swap creates a violation of the restriction on the number of split loads per route. The local improvement of switching origins or destinations within the route is again performed. If the length of the modified route is less than the maximum route length, the swap is accepted. Loads are further swapped within each route until no cost improvement is found.

### 3.2.6 Inter-Route Load Insertion

This procedure is similar to that of intra-route load insertion, except loads are inserted from one route to another. Again, insertion involves moving the entire load to a different route. The load insertion is evaluated to determine if capacity constraints are violated, if the insertion has already been attempted, or if the insertion would return a split load back to its original position. Figure 9 presents an example of a possible load insertion. In this situation, the destination for loads $C$ and $D$ is the same. Therefore, inserting $D$ into route 2 does not require the insertion of an additional destination. This is highlighted to note one of the primary differences between modifications in the PDP and in the VRP. A load may be inserted into a route in the PDP, but the route will not necessarily be altered at all.

The cost of each feasible insertion is determined. If at least one insertion reduces the overall cost, the insertion that results in the maximum reduction in cost is selected. The routes are tested to determine if the insertion creates a violation of the restriction on the number of split loads per route. The local improvement of switching origins or destinations within the route is again performed. If the length of both of the modified routes is less than the maximum route length, the insertion is accepted. Loads are further inserted within each route until no cost improvement is found.

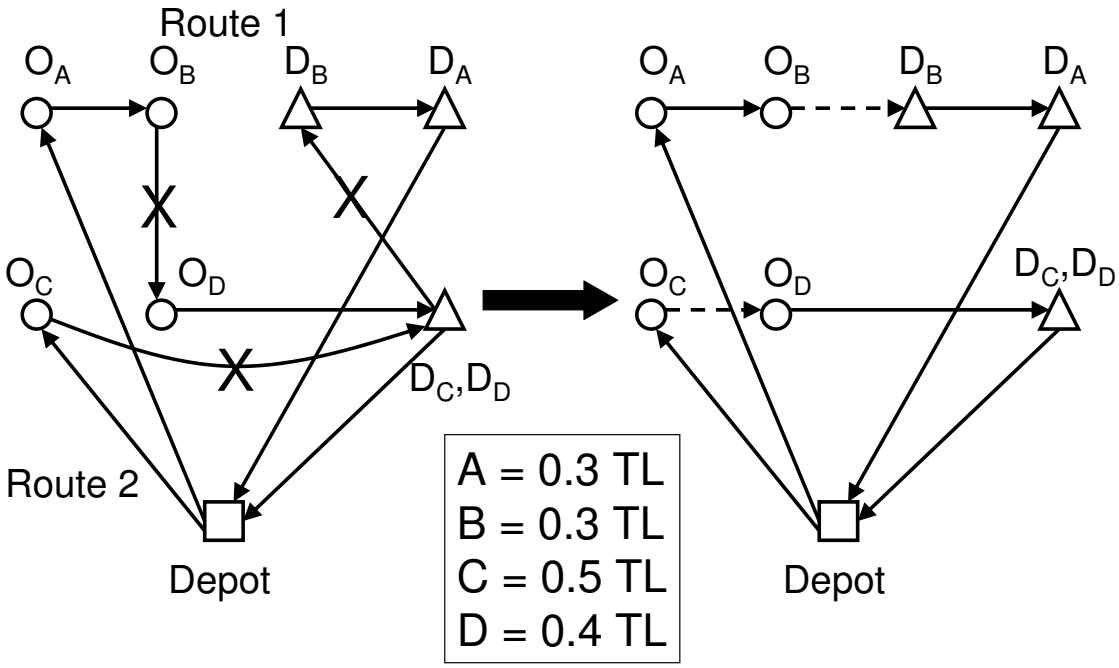


**Figure 9:** Inter-route load insertion example with load $D$ inserted into route 2.

### 3.2.7 Inter-Route Load Swapping

This procedure is similar to that of intra-route load swapping, except loads are swapped between separate routes. Two loads from different routes are selected to be potentially swapped. The swap of the two loads is evaluated to determine if capacity constraints are violated, or if the swap has already been attempted. The potential swaps are the same as those swaps within routes. Figure 10 presents a potential load swap. This is a similar situation as that in the example for the inter-route insertion, in that the destination for loads $C$ and $D$ is the same. Swapping loads $B$ and $D$ does not require an additional destination to be visited on route 1. Rather, the origin and destination for load $B$ are eliminated and only the origin for load $D$ must be added.

The cost of each feasible swap is determined. If at least one swap reduces the overall cost, the swap of the two loads with the maximum reduction in cost is selected. The routes are again tested to determine if the swap creates a violation of the restriction on the number of split loads per route. The local improvement of switching origins or destinations within the routes is again performed. If the length of both of the modified routes is less than the

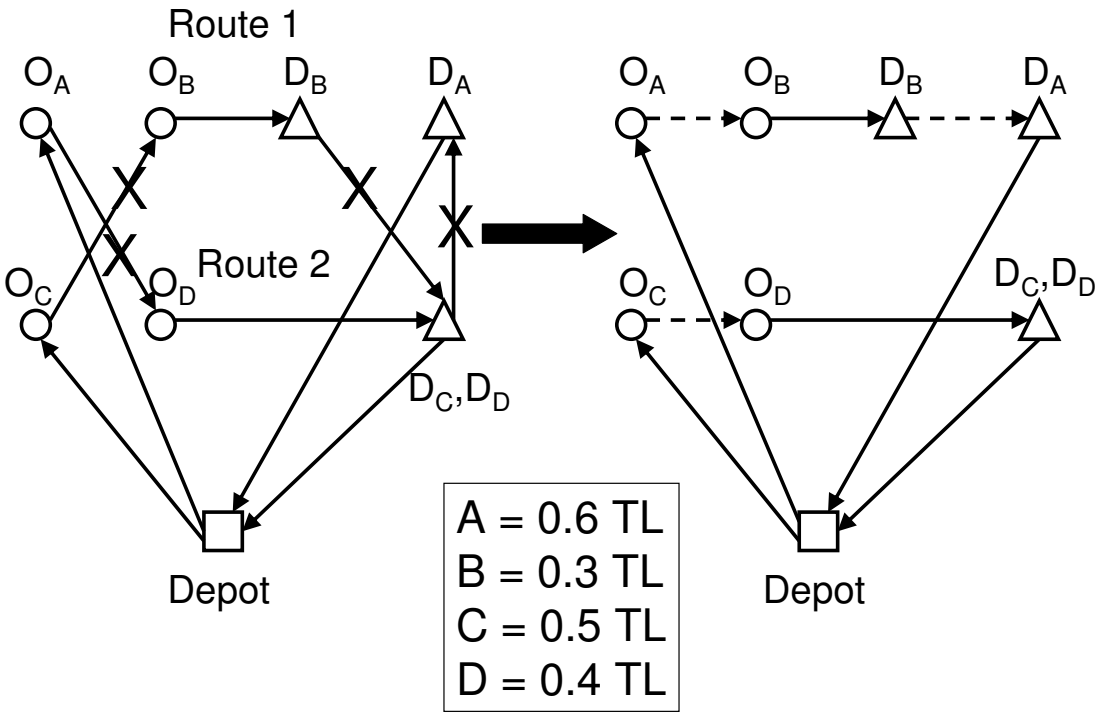


**Figure 10:** Inter-route load swap with loads $B$ and $D$ swapped for each other.

maximum route length, the swap is accepted. Loads are further swapped between routes until no cost improvement is found.

### 3.2.8 Iterations

The iterations of the algorithm are centered around the selection of loads to be split. The first loop of iterations varies the first split load created, while the second loop varies the second split load. That is, a list was maintained of the first split load attempted with the initial solution, and the same first split load was never attempted twice. A similar list was maintained for the second split load selected. Only the first two splits created were tracked in this fashion, as testing showed that there was minimal improvement when the third split load was monitored, while the computational time required to run a similar third iterate loop was significant. A tabu list of the first split load selected is maintained as *tabusplit* and a list of the second split load selected is maintained as *tabusplit*2. A third inner loop of iterations dictates when a solution should be reset. That is, if a solution is not improving

after some time, the heuristic will start again from an earlier solution. Another tabu list, *tabusubsplit*, is maintained for this set of iterations. This list tracks the first split that was attempted after a reset. In this way, the same split load will not be created when the heuristic begins working on an older solution. Finally, a fourth inner loop of iterations runs a solution through the split load creation procedure and the local improvement heuristics multiple times. Please see the following section for a framework of these iterations.

If the solution has improved after finishing all of the iterations of the the fourth inner loop, the iteration counter for the third loop is reset to zero. This gives a solution that is improving more opportunity to reduce costs through the iterative process. The new solution is set as the best current solution, and the split load creation and local improvements continue. If the solution has not improved, the iteration counter for the third loop is incremented, and the heuristic resets with the previous best solution. When the solution is no longer improving, after finishing all iterations of the third inner loop, it is tested against the overall best solution. The iterations continue in this fashion, as different first and second split loads are attempted.

The maximum number of iterations for each loop was determined through testing, balancing cost improvements with degradations in computational time. The number of different first and second split loads tested, or the number of first and second iteration loops, was 10 for each. This resulted in 100 different combinations of split loads tested. The third loop, dictating when a solution should be reset, was run for five iterations. Finally, the fourth loop, searching for local improvements with split load generation, was run for ten iterations.

The order of the local search subalgorithms within the fourth loop of iterations was determined based on the type of local search performed, and the order selected was supported by testing. For example, the intra-route load swap is performed first because it was the most commonly performed improvement. When insertions were performed before swaps, the insertions often duplicated a swap move, as loads were essentially swapped by two insertion moves. This occurred for both inter-route and intra-route moves. Therefore, a reduction in overall iterations for each subalgorithm was found by performing swaps before insertions. Similarly, intra-route moves were duplicated by inter-route moves, so intra-route

moves were performed first.

## 3.3 PDPSL Heuristic Framework

A description of the general heuristic is presented here, followed by the algorithmic subroutines used for split load creation and local improvement. The variables used are as follows:

$iter1$ - counter for the number of iterations for first split load generated

$iter2$ - counter for the number of iterations for second split load generated

$iter3$ - counter for the number of iterations before a restart

$iter4$ - counter for the number of iterations for improvements to be made

$ITERMAIN1$ - upper bound on the number of iterations for $iter1$

$ITERMAIN2$ - upper bound on the number of iterations for $iter2$

$ITERMAIN3$ - upper bound on the number of iterations for $iter3$

$ITERMAIN4$ - upper bound on the number of iterations for $iter4$

$TEMPCOST$ - cost of the solution that is being updated

$BASECOST$ - cost of a solution with no split loads created

$GOODCOST$ - cost of the best known solution for the current first and second split loads created

$BESTCOST$ - cost of the best known solution overall

$WORKCOST$ - cost used for comparison to determine if improvements were made after $iter4$ loop

The general heuristic operates as follows (when a cost is updated, the corresponding solution is updated as well):

**Generate routes:** Create an initial feasible solution and set this as the basic solution, $BASECOST$. Set $GOODCOST$ and $BESTCOST$ equal to $BASECOST$ and $iter1, iter2, iter3$ and $iter4$ equal to 0.

While ($iter1 < ITERMAIN1$)

{

    While ($iter2 < ITERMAIN2$)

    {

Set $TEMPCOST$ equal to $BASECOST$.

While ($iter3 < ITERMAIN3$)

{

Set $WORKCOST$ equal to $TEMPCOST$.

While ($iter4 < ITERMAIN4$)

{

**Create Split Loads.** Update $TEMPCOST$.

If $iter1 > 0$ and $iter4 = 0$, add split load created to *tabusplit*.

If $iter2 > 0$ and $iter4 = 1$, add split load created to *tabusplit*2.

If $iter4 = 0$, add split load created to *tabusubsplit*.

**Combine Routes.** Update $TEMPCOST$.

**Intra-Route Load Swap.** Update $TEMPCOST$.

**Intra-Route Load Insertion.** Update $TEMPCOST$.

**Inter-Route Load Swap.** Update $TEMPCOST$.

**Inter-Route Load Insertion.** Update $TEMPCOST$.

Increment $iter4$.

}

If $TEMPCOST < WORKCOST$, set $GOODCOST$ equal to $TEMPCOST$, and $iter3 = 0$.

If $TEMPCOST \geq WORKCOST$, increment $iter3$ and set $TEMPCOST$ equal to $GOODCOST$.

}

If $GOODCOST < BESTCOST$, set $BESTCOST$ equal to $GOODCOST$.

$iter2 = iter2 + 1$

$iter3 = 0$

}

$iter1 = iter1 + 1$

}

### 3.3.1 Create Split Loads

The following additional lists and variables are used for the algorithm to create split loads:
$insert1done$ - the list of loads that have been previously inserted into a route using the Intra-Route Load Insertion algorithm. This prevents cycling of loads that have just been moved to a spot on a route from being split back to their original location.
$insert2done$ - the list of loads that have been previously inserted into a route using the Inter-Route Load Insertion algorithm. This prevents cycling of loads that have just been moved to a spot on a route from being split back to their original location.
$tabusplitfail$ - the list of loads that were split, but violated the rule on the number of split loads allowed on a route.
$iter5$ - current number of iterations for split load creation.
$ITERINNER5$ - upper bound on the number of iterations to find a split load before continuing with local improvements. After testing, this was set at 10.
The algorithm operates as follows:
While $(iter5 < ITERINNER5)\{$

Select a load to be split and set $A$ equal to the size of the load.

Find excess capacity, $B$, on the same route or another route such that $B < A$.

Split load such that $B$ is moved to fill excess capacity, and $A - B$ is left in original location on route.

Reject if split load on $tabusplit, tabusplit2, tabusubsplit, insertdone1, insertdone2$ or $tabusplitfail$.

Calculate cost of generating split load, $SPLITCOST$.

Accept split load if $SPLITCOST < RANDNUM$, where $RANDNUM$ is a randomly generated number weighted according to cost of the route into which the split load is to be inserted.

If split load creates violation of number of split loads, add to $tabusplitfail$, and

iterate $iter5$.

If split load valid, add to *tabusplit* and update cost, such that $TEMPCOST = TEMPCOST + SPLITCOST$. Reset $tabusplitfail$ and set $iter5$ equal to $ITERINNER5$.

If no split loads are found such that $SPLITCOST < RANDNUM$, set $iter5$ equal to $ITERINNER5$.}

### 3.3.2 Combine Routes

The list $combinefail$ is used for the algorithm that combines routes, tracking those routes that have been previously tested and rejected based on a violation of route length or number of split loads allowed on a route.

The algorithm operates as follows:

Generate all feasible route combinations and determine cost of each combination, $COMBCOST$.

Reject if combination on $combinefail$.

Reject if length of route combination greater than $MAXLENGTH+BUFFER$, where $MAXLENGTH$ is the maximum allowable length of a route and $BUFFER$ is a small value relative to $MAXLENGTH$.

Select combination with minimum $COMBCOST$ if any $COMBCOST$ less than zero.

If combination creates violation of number of split loads add to $combinefail$.

Perform local search for routing improvement, swapping origins or destinations located consecutively on new route.

If length of route combination is greater than $MAXLENGTH$ add to $combinefail$.

If combination is valid, update cost such that $TEMPCOST = TEMPCOST + COMBCOST$. Reset $combinefail$.

Continue until no route combination results in cost improvement.

### 3.3.3 Intra-Route Load Swap

The following additional lists are used for the algorithm that swaps loads within a route:
$swap1fail$ - list of those swaps that have been previously tested and rejected based on a violation of route length or number of split loads allowed on a route.
$swap1done$ - list of those swaps that have been successfully completed. This list is maintained to prevent cycling with the load insertion heuristic.

The algorithm operates as follows:

Generate all feasible load swaps within a single route and determine cost of each swap, $SWAP1COST$.

Reject if swap on $swap1fail, swap1done$ or if swap violates vehicle capacity constraints.

Select swap with minimum $SWAP1COST$, if any $SWAP1COST$ less than zero.

If swap creates violation of number of split loads add to $swap1fail$.

Perform local search for routing improvement, swapping origins or destinations located consecutively on route.

If length of route after swap is greater than $MAXLENGTH$ add to $swap1fail$.

If swap is valid, add to $swap1done$ and update cost such that $TEMPCOST = TEMPCOST + SWAP1COST$. Reset $swap1fail$.

Continue until no load swap results in cost improvement.

### 3.3.4 Intra-Route Load Insertion

The following additional lists are used for the algorithm that inserts loads from one segment of a route to another:
$insert1fail$ - list of those insertions that have been previously tested and rejected based on a violation of route length or number of split loads allowed on a route.
$insert1done$ - list of those insertions that have been successfully completed. This list is maintained to prevent cycling when split loads are generated.

The algorithm operates as follows:

Generate all feasible load insertions within a single route and determine cost of each insertion, $INSERT1COST$.

Reject if insertion on $insert1done, insert1fail$ or $tabusplit$ or if insertion violates vehicle capacity constraints.

Select insertion with minimum $INSERT1COST$, if any $INSERT1COST$ less than zero.

If insertion creates violation of number of split loads, add to $insert1fail$.

Perform local search for routing improvement swapping origins or destinations located consecutively on route.

If length of route after swap is greater than $MAXLENGTH$ add to $insert1fail$.

If insertion is valid, add to $insert1done$ and update cost such that $TEMPCOST = TEMPCOST + INSERT1COST$. Reset $insert1fail$.

Continue until no insertion results in cost improvement.

### 3.3.5 Inter-Route Load Swap

The following additional lists are used for this algorithm:

$swap2fail$ - list of those swaps that have been previously tested and rejected based on a violation of route length or number of split loads allowed on a route.

$swap2done$ - list of those swaps that have been successfully completed. This list is maintained to prevent cycling with the load insertion heuristic.

The algorithm operates as follows:

Generate all feasible load swaps between any two routes and determine cost of each swap, $SWAP2COST$.

Reject if swap on $swap2fail, swap2done$ or if swap violates vehicle capacity constraints.

Select swap with minimum $SWAP2COST$, if any $SWAP2COST$ less than zero.

If swap creates violation of number of split loads add to $swap2fail$.

Perform local search for routing improvement swapping origins or destinations

located consecutively on either route.

If length of either route after swap is greater than $MAXLENGTH$ add to $swap2fail$.

If swap valid, add to $swap2done$ and update cost such that $TEMPCOST = TEMPCOST + SWAP2COST$. Reset $swap2fail$.

Continue until no load swap results in cost improvement.

### 3.3.6 Inter-Route Load Insertion

The following additional lists are used for this algorithm:

$insert2fail$ - list of those insertions that have been previously tested and rejected based on a violation of route length or number of split loads allowed on a route.

$insert2done$ - list of those insertions that have been successfully completed. This list is maintained to prevent cycling when split loads are generated.

The algorithm operates as follows:

Generate all feasible load insertions between routes and determine cost of each insertion, $INSERT2COST$.

Reject if insertion on $insert2done$, $insert2fail$ or $tabusplit$ or if insertion violates vehicle capacity constraints.

Select insertion with minimum $INSERT2COST$, if any $INSERT2COST$ less than zero.

If insertion creates violation of number of split loads, add to $insert2fail$.

Perform local search for routing improvement swapping origins or destinations located consecutively on either route.

If length of either route after swap is greater than $MAXLENGTH$ add to $insert2fail$.

If insertion valid, add to $insert2done$ and update cost such that $TEMPCOST = TEMPCOST + INSERT2COST$. Reset $insert2fail$.

Continue until no insertion results in cost improvement.

## *3.4 Conclusions*

In this chapter, a heuristic for the solution of the PDPSL was presented. This heuristic combines elements from several methods, using a technique similar to tabu search for split load generation, followed by local search improvements such as route combination, load swaps and load insertions. The method for generating split loads was described, along with the various subalgorithms used in the PDPSL Heuristic. Finally, a framework for the heuristic and each of the subalgorithms was presented.

# CHAPTER IV

# COMPUTATIONAL EXPERIENCE REGARDING THE PDPSL HEURISTIC

The following chapter presents the results of numerical experiments that are designed to indicate the potential benefit of using split loads. The heuristic described in Chapter 3 is used to address the following questions:

1. What is the cost benefit of allowing the use of split loads?

2. How much benefit is lost if routes are limited to carrying at most one split load?

The latter question is of interest due to the reduction in computational complexity that results when the number of split loads per route is limited. In Section 4.1 the design of the experiment used to evaluate split loads is described, while Section 4.2 presents the results of the experiment. Section 4.3 provides a benchmark for the heuristic by comparing it to the mixed integer program described in Section 2.4. Conclusions are presented in Section 4.4.

## *4.1 Experimental Design*

The heuristic was tested on problem sets of three sizes, with 75, 100, or 125 transportation requests. Each transportation request contains the origin and destination location information, and the fraction of a truckload to be delivered. Coordinates for the pickup and delivery locations were randomly generated with a uniform distribution over the range [-40,40] for both X and Y coordinates. The results below are reported with the location of the depot fixed at [0,0] for each data set. Testing with the depot moved to other locations did not alter the results significantly. Each problem size has five origins from which loads could be picked up. The 75 request problem then has 15 destination locations, the 100

request problem has 20 destination locations, and the 125 request problem has 25 destination locations. Testing with a different number of origins found no meaningful changes to the results. Every origin-destination combination has a load to be delivered between the two locations, such that the load matrix has all non-zero values. Three configurations with different origin and destination locations were generated for each problem size.

The load sizes were also randomly generated. The sizes were all less than or equal to vehicle capacity. This was done in order to determine the load sizes that can completely fit onto a vehicle that provide the most benefit from split loads. The following chapter presents tests run with load sizes greater than vehicle capacity. Without loss of generality, vehicle capacity was fixed at 1. In order to provide insight into the problem, the load sizes were generated within a variety of ranges. Two different range groupings were used to determine these sizes. The first set of groupings contained loads generated within narrow ranges of size. Eight ranges were used to bound the load sizes, with five different sets of load sizes generated for each range. The ranges indicated the upper and lower bound (inclusive of the bound) on the fraction of the vehicle capacity that the load can occupy, and they were [0.11-0.2], [0.21-0.3], [0.31-0.4], [0.41-0.5], [0.51-0.6], [0.61-0.7], [0.71-0.8], and [0.81-0.9]. Load sizes below 0.1 or above 0.9 were not used as splits of these loads rarely occur and any benefit was negligible. The load sizes were randomly generated over each range with a uniform distribution. Each of the three location configurations was matched with each of the five load sets within a load range, resulting in 15 different problems for each load range, and 120 problems overall.

The second set of groupings contained loads generated within wider ranges of size. Four ranges were used to bound the load sizes, with 10 different sets of load sizes for each range. The ranges were [0.1-0.6], [0.6-1.0], [0.3-0.6], and [0.1-1.0]. The load sizes were randomly generated over each range with a uniform distribution. Each of the three location configurations was matched with each of the ten load configurations, resulting in 30 different problems for each load range, and 120 problems overall.

The heuristic was used to solve each problem under two scenarios, both with and without split loads. The heuristic with split loads followed the procedure described earlier in

this chapter. The heuristic without split loads omitted the split load generation step and iterated through the local improvement steps three times. Testing showed that no additional improvements could be made with the heuristic to the routes after three iterations. An additional scenario was also tested, involving the restriction that only one split load is allowed per route. While it has been proven that there is only one split load per route for the Split Delivery Vehicle Routing Problem [58], this can not be shown for the PDPSL. However, through experimentation, we show that the PDPSL does exhibit this behavior under certain conditions.

An additional constraint tested with all of the problem sets was the implementation of a route length restriction. The Vehicle Routing Problem has an inherent route length restriction that is determined by vehicle capacity. The PDP does not have this characteristic as capacity can be continually replenished whenever the vehicle has emptied all loads picked up. While the optimal solution would result in the vehicle traveling continuously without returning to the depot until all loads have been serviced, a vehicle could not do this in real life. Therefore, a route length restriction was imposed requiring that the vehicle return to the depot after traveling some maximum distance. Based on current U. S. Department of Transportation rules on driver hours and the expected distance traveled within the maximum time allowed, this distance was set at 500 miles. The experiments below indicate the effect this maximum distance has on the various results. All experiments were run on a 2.4 GHz Xeon processor with a 400Mhz frontside bus and 2 GB RAM.

## *4.2 Experimental Results*

We initially discuss the cost benefit found when using split loads. The use of the narrow load ranges allows for an accurate depiction of the specific load sizes that provide the most (or least) benefit. The wider load ranges show the benefit that may be found in a real world setting in which load sizes are more varied. The discussion of cost benefit is followed by an analysis of restricting the number of split loads per route. The tradeoff between the resulting improvement in computational time and the reduction in cost benefit is quantified.

### 4.2.1 Benefit of Split Loads

Figures 11, 12, and 13 present the percentage increase in cost when split loads are not allowed for the 75, 100, and 125 request problem sets with narrow load size ranges, respectively. Some cost benefit is found with the use of split loads for every load range. However, more benefit is clearly found with certain load sizes. The results for each problem set support a main theoretical result from Chapter 2. That is, the most significant cost benefit with split loads is found when the load sizes are just above 1/2 of vehicle capacity. When splitting is allowed with these load sizes, full truckloads are created through the combination of 1/2 truckloads, with the remainder of the loads delivered on an additional route. The cost benefit becomes almost negligible in the range 0.41-0.5. This is because loads are easily combined on a vehicle without any splitting needed. Furthermore, the loads are large enough that when they are combined on a vehicle, there is little room for a split load to be inserted. The benefit slightly increases in the range 0.31-0.4, as loads can be combined on a vehicle with more room for splits to be inserted. The benefit then continues to decline as load sizes become smaller, with a greater likelihood that multiple loads can be combined on a vehicle at the same time with no splitting required.

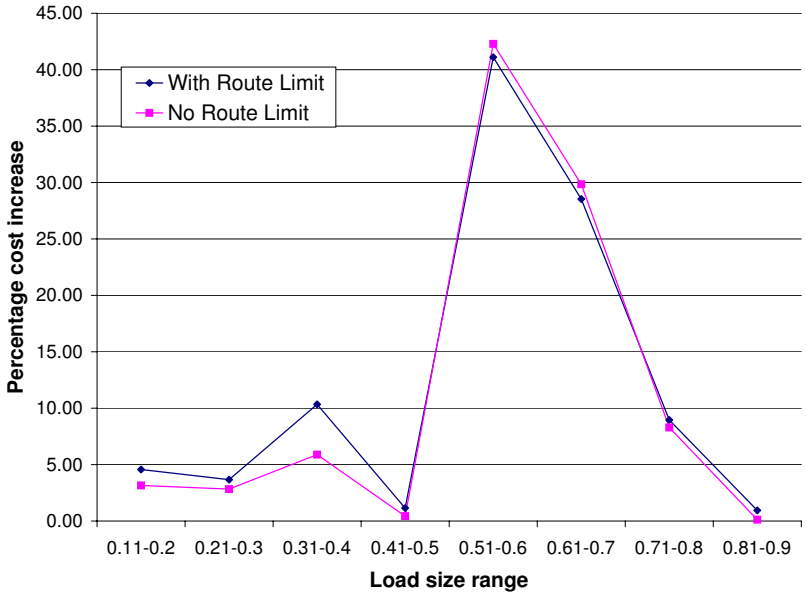


**Figure 11:** Average percentage cost increase without split loads for each load range tested for the 75 request problem set.

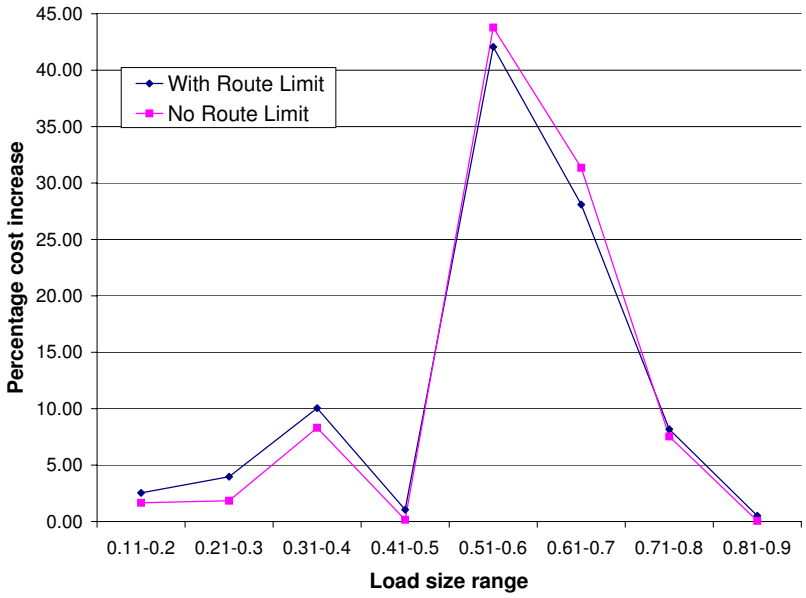


**Figure 12:** Average percentage cost increase without split loads for each load range tested for the 100 request problem set.

The results are slightly different for those load ranges greater than 0.51-0.6. The range 0.61-0.7 has the second highest cost benefit as multiple loads can not be combined on a vehicle at any time without split loads. However, the benefit is lower than for the range 0.51-0.6 because there is less capacity available for a split load when a full load is already on a vehicle. Therefore, when part of a load is placed on a vehicle with a full load, there are fewer options for the remainder of the load due to its greater size. The decrease in benefit becomes more dramatic as the load sizes increase, with a benefit of less than 10% for the range 0.71-0.8 and almost no benefit for the range 0.81-0.9.

The trends are very similar for both scenarios with and without a route limit. This is primarily due to the fact that the depot is located within close proximity of the origin and destination locations. The cost of travel to the depot when a route is completed is not significant. Therefore, there is little benefit to decreasing the number of routes by eliminating the length restriction. Limited testing has shown that if the depot is placed much further from the other locations, the difference between the two scenarios becomes more defined. However, the depot is generally located within close proximity to the origins and destinations in most real life problems. The study of depot location and route length

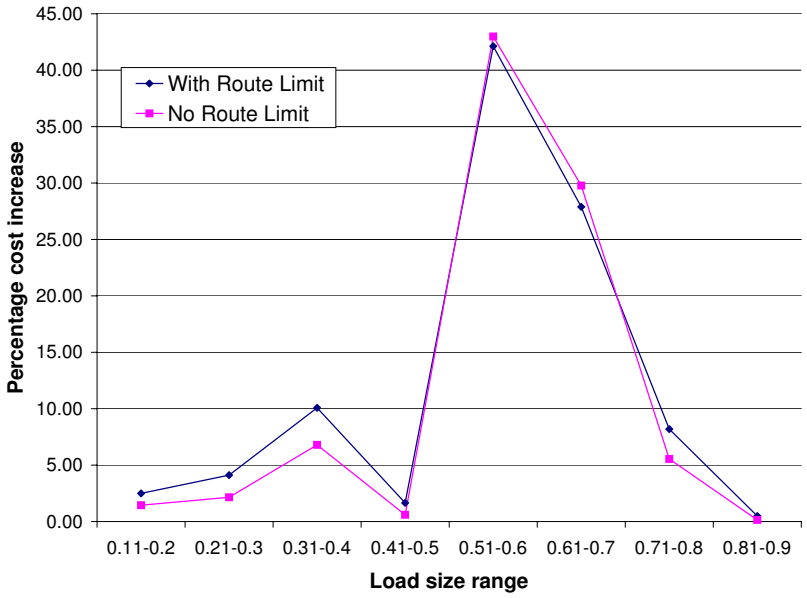


**Figure 13:** Average percentage cost increase without split loads for each load range tested for the 125 request problem set.

restrictions is beyond the scope of this paper, and we do not present further analysis on this topic.

Table 1 presents the average CPU time, in minutes, required to find the solutions reported and the average number of iterations for which the heuristic performs the local improvement steps. The counter for the iterations is incremented each time the heuristic completes the split load creation step, prior to beginning the load combination subalgorithm (and continuing on to the other local improvement subalgorithms).

The CPU time increases significantly with problem size for every load range, doubling with an increase of 25 service requests in most cases. However, the number of iterations does not reflect this same increase. In fact, in some instances the number of iterations actually decreases with an increase in problem size. These instances occur for those load ranges that find the least cost benefit from using split loads. Because the heuristic can not find improvements with split loads, it completes close to the minimum number of iterations. This minimum number is the same for each problem size, so that all problems result in a similar number of iterations. Conversely, the largest increase in iterations occurs for those

**Table 1:** Average CPU time in minutes and average number of iterations for solution with split loads for narrow load ranges.

| | Problem size | 75 | | 100 | | 125 | |
|---|---|---|---|---|---|---|---|
| | Load range | No route limit | Route limit | No route limit | Route limit | No route limit | Route limit |
| Average CPU time (minutes) | 0.11 - 0.2 | 3.4 | 3.8 | 6.7 | 9.3 | 9.5 | 17.3 |
| | 0.21 - 0.3 | 6.8 | 5.3 | 7.5 | 12.7 | 18.7 | 22.7 |
| | 0.31 - 0.4 | 12.4 | 10.3 | 19.7 | 24.4 | 38.1 | 48.1 |
| | 0.41 - 0.5 | 3.6 | 3.6 | 6.1 | 8.0 | 9.5 | 13.8 |
| | 0.51 - 0.6 | 58.2 | 35.9 | 155.4 | 90.8 | 286.3 | 175.6 |
| | 0.61 - 0.7 | 30.3 | 21.5 | 88.7 | 49.1 | 164.6 | 92.1 |
| | 0.71 - 0.8 | 11.8 | 11.7 | 24.0 | 21.6 | 28.5 | 34.4 |
| | 0.81 - 0.9 | 5.5 | 4.2 | 9.5 | 8.6 | 18.8 | 13.9 |
| Average number of iterations | 0.11 - 0.2 | 1088 | 956 | 1012 | 959 | 845 | 969 |
| | 0.21 - 0.3 | 1109 | 1106 | 1047 | 1305 | 1364 | 1453 |
| | 0.31 - 0.4 | 1690 | 2205 | 2336 | 2730 | 2392 | 3326 |
| | 0.41 - 0.5 | 838 | 871 | 921 | 857 | 811 | 953 |
| | 0.51 - 0.6 | 5575 | 6422 | 7089 | 8385 | 7987 | 9907 |
| | 0.61 - 0.7 | 4315 | 4792 | 5465 | 6000 | 5982 | 7026 |
| | 0.71 - 0.8 | 1773 | 2670 | 2029 | 2787 | 1840 | 2930 |
| | 0.81 - 0.9 | 1043 | 1038 | 1138 | 1158 | 1147 | 1181 |

load ranges with the most potential cost benefit with split loads.

As the number of iterations at worst increases linearly with problem size, the increase in computational time is attributed to the greater number of steps that must be performed within each subalgorithm to find an improvement. As problem size increases, the number of combinations that can be tested for local improvement is significantly increased. This is compounded within each subalgorithm. Furthermore, an additional result of the linear increase in the number of iterations is a slight decrease in the cost benefit as the problem size increases. This is not very evident for the narrow set of load size ranges; however, it will become more pronounced in the following discussion regarding the wider set of load range sizes.

Figures 14, 15, and 16 present cumulative histograms of the number of problems versus the percentage increase in cost when split loads are not allowed for the 75, 100, and 125 request problem sets with wider load size ranges, respectively. These histograms are constructed in such a way that the farther a point is to the right on the graph, the more problems with a higher cost increase. Therefore, the lines farthest to the right indicate those load ranges with the most cost benefit from split loads. The graphs presented are different from the narrow load ranges as the wide load ranges can not be easily adapted to a similar format. The histograms provide similar information for different problem specifications.

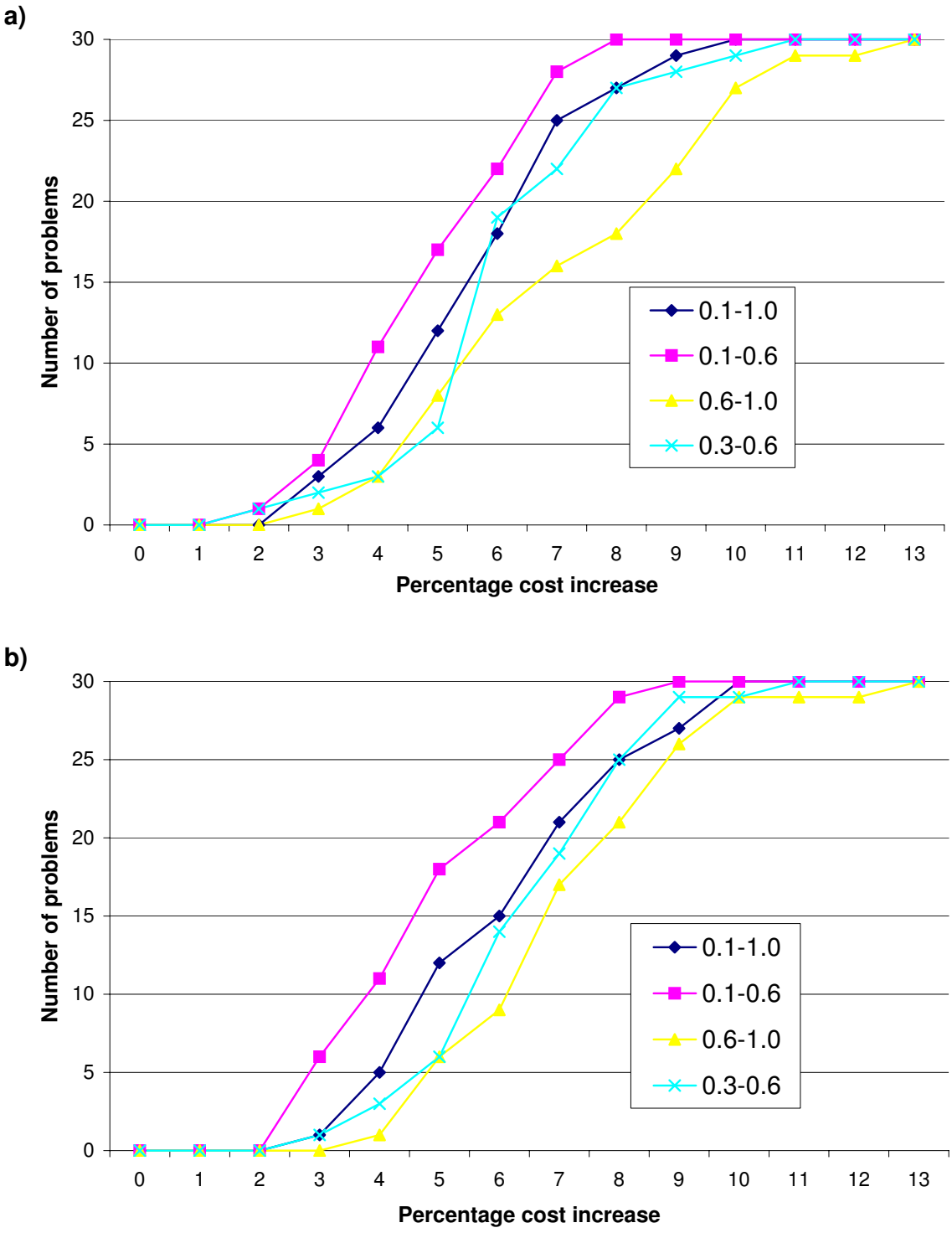


**Figure 14:** a) Cumulative histogram for the number of problems versus the percentage cost increase without split loads for 75 request problem set with no route length limit and b) with route length limit.

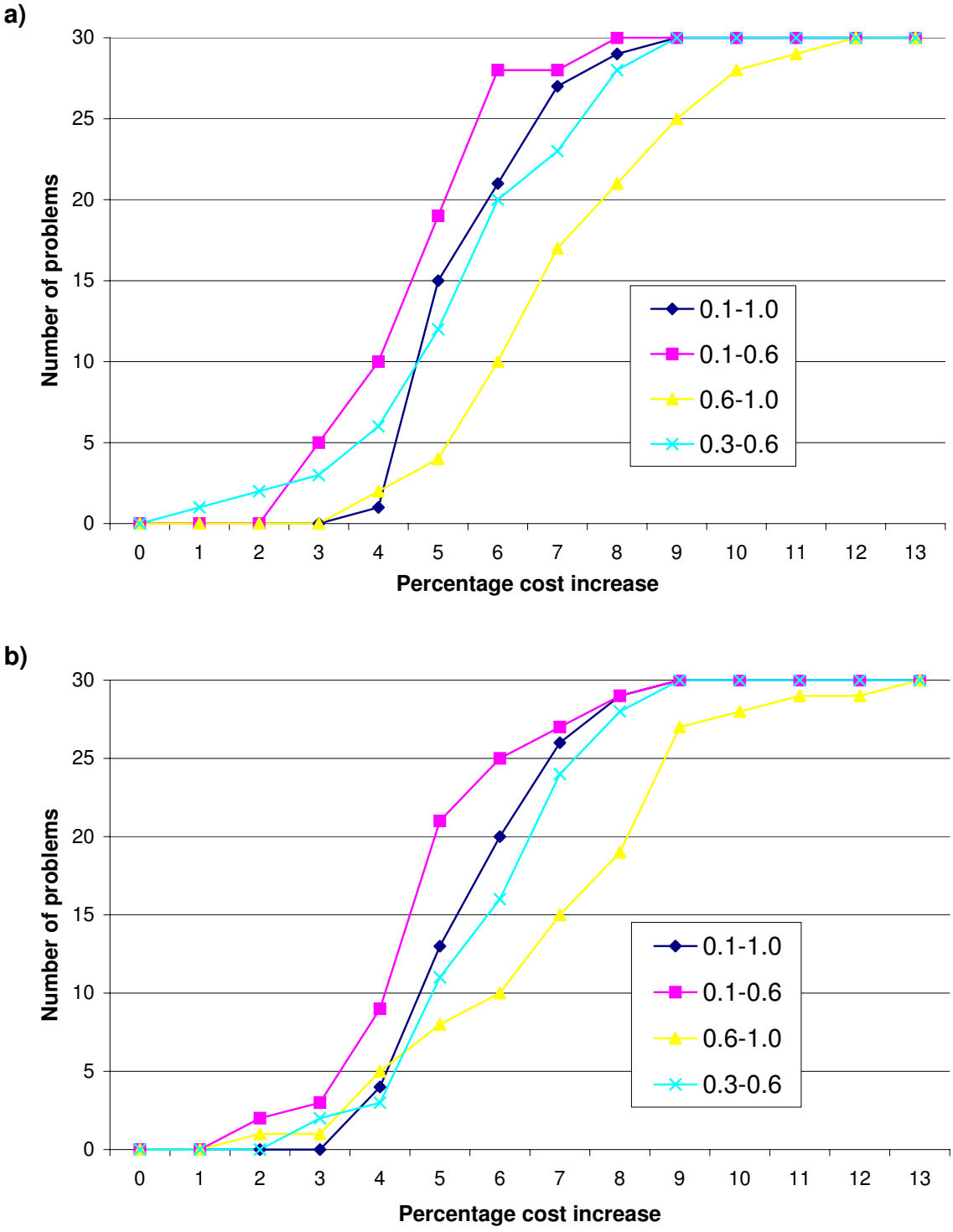


**Figure 15:** a) Cumulative histogram for the number of problems versus the percentage cost increase without split loads for 100 request problem set with no route length limit and b) with route length limit.

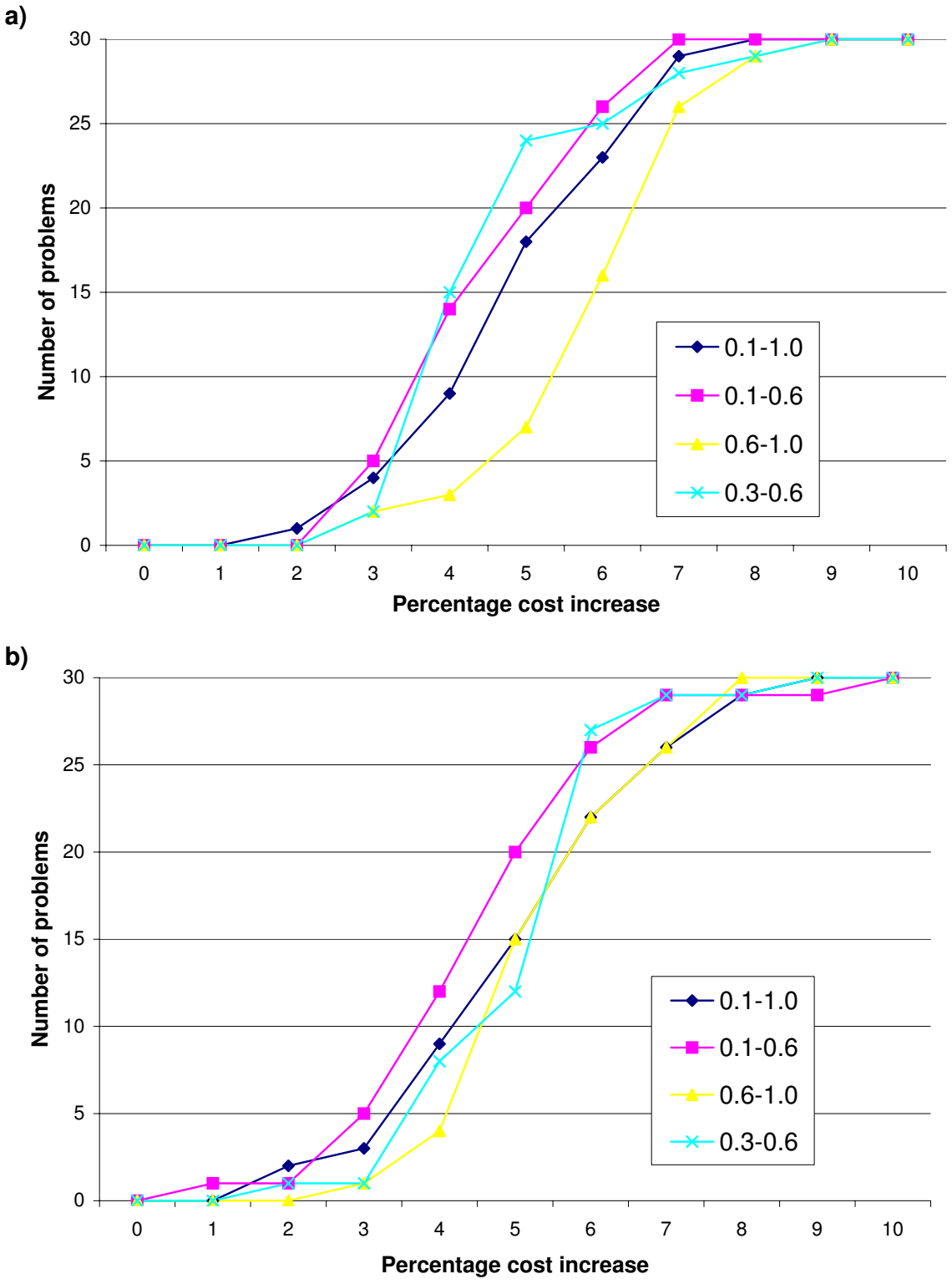


**Figure 16:** a) Cumulative histogram for the number of problems versus the percentage cost increase without split loads for 125 request problem set with no route length limit and b) with route length limit.

Again, as with the narrow load size ranges, there is a cost benefit to the use of split loads for every load size range. Although the benefit is not as marked as for some of the narrow ranges, some problems were found to have a cost increase upwards of 13 %. Also, the results regarding the most beneficial load sizes support those found with the narrow load size ranges. The range 0.6-1.0 finds the most benefit in every situation. Also, the ranges 0.3-0.6 and 0.1-1.0 are comparable in terms of cost benefit, but slightly lower than the range 0.6-1.0. This indicates that while the most benefit can be found with loads greater than 1/2 vehicle capacity, benefit can also be found when load sizes vary over the range of available capacity. The range 0.1-0.6 found the least benefit, as the smaller loads are less likely to be split.

Again, there is little difference between those instances with and without a route limit. However, there is some difference between the different problem sizes. The cost benefits are most varied between the load ranges for the 75 and 100 request problem sets. That is, there is markedly more benefit in the range 0.6-1.0 then in the other ranges. This is less obvious for the 125 request problem set. Also, there is slightly less benefit with split loads overall for the 125 request set. The construction of the heuristic is the primary factor behind this. Because the heuristic is run for a limited number of iterations, the solution space of a larger problem may not be explored as extensively as a smaller problem. Increasing the number of iterations improves the solution, but at a very significant cost in computational time.

Table 2 further emphasizes the results found in the histograms, as it provides the average percentage cost increase without split loads for each data set, the average CPU time required to obtain the solution, and the average number of iterations. Clearly, the benefit of split loads decreases for the 125 request problem size over all load ranges. The most significant drop is in the two ranges, 0.6-1.0 and 0.3-0.6, with more benefit found with the smaller problem sizes. The results indicate that while 125 request problems require almost double the computational time, the number of iterations marginally increases, or even drops. The number of iterations is a direct indicator of how successful the heuristic is in making improvements to the solution. As long as cost savings are found, the heuristic will continue without interruption. However, if after some time no improvements are

**Table 2:** Average percentage cost increase without split loads, average CPU time (in minutes) for solution with split loads, and average number of iterations for solution with split loads for wide load ranges.

| | Problem size | 75 | | 100 | | 125 | |
|---|---|---|---|---|---|---|---|
| | Load range | No route limit | Route limit | No route limit | Route limit | No route limit | Route limit |
| Average % cost increase | 0.1 - 1.0 | 5.6 | 5.9 | 5.4 | 5.4 | 4.7 | 5.0 |
| | 0.1 - 0.6 | 4.8 | 4.8 | 4.5 | 4.7 | 4.3 | 4.5 |
| | 0.6 - 1.0 | 6.9 | 6.8 | 7.0 | 6.6 | 5.8 | 5.2 |
| | 0.3 - 0.6 | 5.9 | 6.4 | 5.3 | 5.7 | 4.4 | 4.9 |
| Average CPU minutes | 0.1 - 1.0 | 8.9 | 6.4 | 22.6 | 14.6 | 40.5 | 25.9 |
| | 0.1 - 0.6 | 6.3 | 5.8 | 13.5 | 13.2 | 24.6 | 25.1 |
| | 0.6 - 1.0 | 11.0 | 7.9 | 24.8 | 16.7 | 40.1 | 22.6 |
| | 0.3 - 0.6 | 8.1 | 7.5 | 17.9 | 16.1 | 29.5 | 28.5 |
| Average number of iterations | 0.1 - 1.0 | 1905 | 1817 | 2285 | 2246 | 2494 | 2525 |
| | 0.1 - 0.6 | 1524 | 1412 | 1674 | 1611 | 1815 | 1868 |
| | 0.6 - 1.0 | 1773 | 2043 | 2095 | 2375 | 2083 | 2108 |
| | 0.3 - 0.6 | 1794 | 1790 | 2010 | 2005 | 2020 | 2204 |

found, the heuristic ends one search to begin another, decreasing the number of iterations. Exploring the solution space of the 125 request problem is exponentially more difficult than exploring the space of the smaller problem sizes, and, therefore, the possibility of finding cost improvement decreases.

### 4.2.2 Limiting the Number of Split Loads

It has been shown that there exists an optimal solution of the Split Delivery Vehicle Routing Problem in which every route has at most one split load [58]. However, this same result does not hold for the PDPSL. Figure 17 presents a simple example of a situation in which this does not hold. The vehicle visits the origin for load A first and picks up the full load.

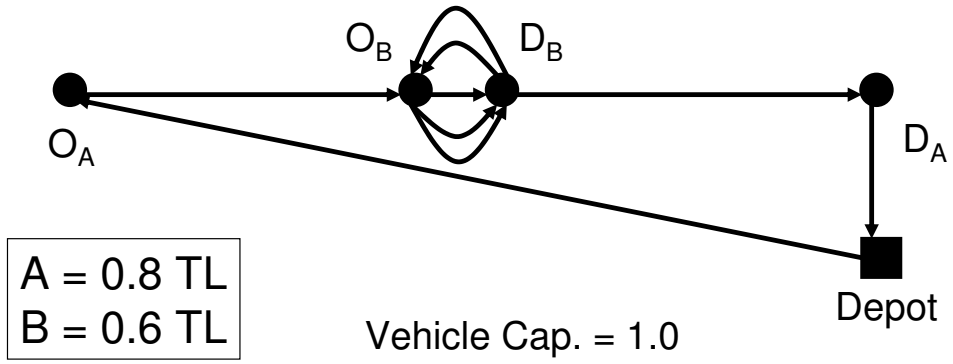


**Figure 17:** Example in which more than one split load may occur on a route.

Without split loads, the vehicle would be required to deliver the load and then return to deliver load B. With split loads, the vehicle can continue on to deliver load B while load A is on the vehicle. When the vehicle arrives at the origin for B, the available capacity allows for load B to be delivered in three segments of 0.2 each. In this way, two split loads are delivered on the route.

Limiting the number of split loads per route decreases the size of the solution space to be explored, thereby reducing the computational investment required. However, this limit may also decrease the quality of the solution. It is of interest to determine the amount by which both the computational time and solution quality decrease when a limit is placed on the number of split loads per route.

As described in Section 3.2.2, any modification to a route performed by the heuristic may be rejected if it creates more split loads on the route than some fixed parameter. For all other testing, this parameter was set as a very large number so as to not limit the number of split loads per route. However, for this analysis, the parameter was set at 1. The heuristic was applied to the same set of problem instances as for the previous testing. Tables 3 and 4 present the percentage amount that the cost increased compared to the results presented in the previous section when a split load limit was implemented. The resulting percentage

**Table 3:** Average percentage cost increase and CPU time decrease with one split load limit for narrow load ranges.

| | Problem size | 75 | | 100 | | 125 | |
|---|---|---|---|---|---|---|---|
| | Load range | No route limit | Route limit | No route limit | Route limit | No route limit | Route limit |
| % cost increase with split limit | 0.11 - 0.2 | 2.1 | 1.1 | 0.6 | 0.1 | 0.2 | 0.0 |
| | 0.21 - 0.3 | 2.0 | 1.1 | 1.2 | 0.5 | 0.2 | 0.8 |
| | 0.31 - 0.4 | 4.6 | 3.8 | 6.8 | 3.7 | 6.0 | 3.9 |
| | 0.41 - 0.5 | 0.2 | 0.1 | 0.3 | 0.1 | 0.3 | 0.1 |
| | 0.51 - 0.6 | 25.6 | 20.6 | 27.7 | 22.1 | 27.9 | 22.8 |
| | 0.61 - 0.7 | 15.8 | 11.7 | 17.5 | 11.3 | 17.6 | 11.8 |
| | 0.71 - 0.8 | 7.1 | 3.6 | 6.4 | 3.4 | 4.5 | 3.2 |
| | 0.81 - 0.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 |
| Average % CPU time decrease | 0.11 - 0.2 | 70.5 | 33.5 | 72.0 | 21.6 | 43.1 | 18.0 |
| | 0.21 - 0.3 | 72.1 | 30.9 | 72.9 | 31.0 | 44.3 | 28.2 |
| | 0.31 - 0.4 | 75.9 | 72.2 | 91.4 | 71.1 | 88.2 | 73.7 |
| | 0.41 - 0.5 | 71.6 | 19.3 | 67.4 | 6.0 | 52.1 | 8.1 |
| | 0.51 - 0.6 | 97.6 | 94.8 | 98.0 | 95.0 | 98.1 | 94.9 |
| | 0.61 - 0.7 | 95.4 | 90.6 | 96.2 | 90.3 | 96.7 | 89.6 |
| | 0.71 - 0.8 | 54.9 | 50.7 | 30.2 | 37.0 | 53.1 | 32.8 |
| | 0.81 - 0.9 | 50.3 | 2.5 | 8.3 | 1.7 | 18.7 | 1.4 |

**Table 4:** Average percentage cost increase and CPU time decrease with one split load limit for wide load ranges.

| | Problem size | 75 | | 100 | | 125 | |
|---|---|---|---|---|---|---|---|
| | Load range | No route limit | Route limit | No route limit | Route limit | No route limit | Route limit |
| % cost | 0.1 - 1.0 | 2.7 | 1.2 | 2.6 | 0.8 | 2.6 | 0.6 |
| increase | 0.1 - 0.6 | 1.8 | 0.6 | 1.5 | 0.6 | 1.9 | 0.6 |
| with split | 0.6 - 1.0 | 4.5 | 1.5 | 4.7 | 1.4 | 4.0 | 0.9 |
| limit | 0.3 - 0.6 | 2.2 | 1.2 | 2.1 | 0.7 | 1.9 | 0.7 |
| Average | 0.1 - 1.0 | 84.4 | 41.4 | 87.0 | 40.6 | 86.8 | 36.3 |
| % CPU | 0.1 - 0.6 | 73.9 | 34.4 | 75.6 | 29.2 | 77.3 | 28.7 |
| time | 0.6 - 1.0 | 70.2 | 26.1 | 73.1 | 21.5 | 70.9 | 6.6 |
| decrease | 0.3 - 0.6 | 81.4 | 49.5 | 82.1 | 44.4 | 82.2 | 39.7 |

decrease in CPU time is also presented.

The resulting increase in cost from limiting the number of split loads per route corresponds directly to the cost benefit of using split loads for the problems with a narrow load size range. Those load ranges with the most benefit of split loads find the highest increase in cost when the use of split loads is limited. Similarly, those load ranges with little benefit from split loads resulted in almost identical, if not identical, solutions with and without a limit on the number of split loads. The decrease in computational time followed this pattern as well. Limiting the number of split loads per route reduces the feasible solution space to be explored. This reduction is most evident in those instances where there is the most potential for split loads, as in the ranges greater than one half vehicle capacity.

The decrease in computational time is much more pronounced when there is no route limit imposed. This is because the split load limit acts as somewhat of a route limit. As a route can only have one split load, and split loads generate cost savings, the heuristic will avoid combining routes to allow for more split loads to be used. Without a split load limit, the heuristic builds one long route for every problem solution. This is to avoid the additional cost of having the vehicle return to the depot at the beginning and end of a route. However, with a split load limit, the additional cost of returning to the depot is outweighed by the benefit of being able to use a split load on multiple routes. The resulting decrease in computational time is because there are fewer beneficial route combinations to explore.

The results regarding the percentage increase in cost for the wider load ranges shown in Table 4 are similar to those found for the narrow load ranges. The increase is most

significant for those ranges with the most benefit from split loads. However, as was evident without the split load limit, the cost differences for these ranges are not quite as dramatic. This indicates that when the load sizes are spread over a wider range, limiting the number of split loads does not significantly affect solution quality. The percentage decrease in computational time does not necessarily follow the findings for the narrow load ranges. In this situation, the ranges with the most benefit from split loads find the least decrease in time. When the split load limit is utilized, the computational times for the wider load size ranges are very similar. This was not true for the narrow load ranges. Because the 0.6-1.0 load size range had the highest computational time without the limit, the least percentage decrease resulted. The computational times were similar for each load range after implementation of the split limit because the benefit for split loads is similar for each range.

## *4.3 Evaluation of Heuristic*

We now use the mixed integer program described in Section 2.4 to provide an evaluation of the heuristic relative to an exact solution method. The MIP was consistently able to solve to optimality problems with up to six requests. Certain larger problems could be solved in some instances, but not often enough to allow for thorough testing. Therefore, the size of the problems for testing was set at six requests, with two origins and three destinations. The data was randomly generated similarly to the data for the heuristic testing above. The load ranges used were [0.1-0.6] and [0.3-0.6], as the program could not consistently solve problems with larger load sizes. Ten load configurations were generated for each range with two location configurations, resulting in 20 problems for each load range.

The MIP was used to solve each problem, as was the heuristic. Information regarding the average decrease in cost found when using the exact formulation, along with the computational time required for the MIP, is found in Table 5. The average cost decrease in each case indicates that the heuristic does have some potential for improvement. While the heuristic obtained the exact solution in the majority of cases, some problems resulted in an almost 8% cost decrease. In many instances this was because the heuristic was stuck

**Table 5:** Average percentage cost decrease and CPU time for exact MIP formulation.

| Load range | 0.1 - 0.6 | | 0.3 - 0.6 | |
|---|---|---|---|---|
| Location set | 1 | 2 | 1 | 2 |
| Average % cost decrease | 2.37 | 2.75 | 1.90 | 2.62 |
| Average CPU time (min) | 58.83 | 12.33 | 363.40 | 600.17 |

in a local optimum that could only be improved by changing the ordering of entire loaded segments (those segments on which the vehicle is empty at beginning and end). The heuristic does not swap full loaded segments on a route. For example, assume a route has two loaded segments $A$ and $B$, in which each segment is comprised of more than one origin and one destination. If the order is currently $AB$, the heuristic will not test for the order $BA$ (unless it comes upon this solution through alternate local improvements). This test was added to the heuristic, resulting in some improved solutions for the small problem instances. However, when tested on the data sets with 75 service requests, negligible cost improvements were found with significant increases in computational time. Furthermore, because the load sizes were mainly below one half vehicle capacity, many route plans did not contain split loads. Therefore, these tests are more representative of the quality of the routing mechanism of the heuristic, rather than the performance of the split load selection.

The computational times required by the MIP are clearly excessive. Note that no attempt was made to exploit the structure of the MIP with specialized solution approaches, since solution procedures for the MIP has not been the focus of this research. Specialized solution approaches would certainly reduce the reported CPU times. However, the need for a specialized approach even for such a small instance indicates the complexity of the problem.

## *4.4 Conclusions*

The heuristic was tested on randomly generated data sets with load sizes falling in several ranges, quantifying the benefit of using split loads. The experimental results supported the structural results reported in Chapter 2, as the most benefit was found with loads

just over one half vehicle capacity. Most other load ranges found some benefit, with the wider ranges of loads showing an average benefit of up to 7%. Further tests were run to determine the effect of limiting the number of split loads allowed per route. This limit was found to be computationally beneficial, with minimal loss in solution quality for the wider load ranges. Finally, the heuristic was compared to an exact solution method, testing very small problem instances. These results indicated that even small instances of the PDPSL require a significant amount of time to find an exact solution, and meaningful instances of the problem can not be solved to optimality. Also, while the heuristic performed well on average, some instances showed that there is the potential for some improvement.

# CHAPTER V

# FEDERAL EXPRESS CASE STUDY

This chapter presents a case study on Federal Express in which the heuristic presented in Chapter 3 is adapted and applied to a real data set. The heuristic is used to determine the benefit of using split loads in a real world problem. Also, it will determine what the benefit is for pooling loads that are serviced by several distinct fleets and servicing them with one united fleet. Section 5.1 discusses the historical background of Federal Express and indicates the motivation for this case study. Much of the historical discussion is adapted from Hoffman, Hough and Nowak [47]. Section 5.2 describes the problem to be analyzed, while Section 5.3 presents the design of the experiments. Sections 5.4 and 5.5 present and interpret the various results of the computational study. Finally, Section 5.6 summarizes the results of the case study.

## *5.1 Company Background*

The package express industry in the United States has changed dramatically from its origins on bicycles on the streets of Seattle to the internet driven supply chain of the industry today. With close to $44 billion in total global revenues, the biggest player in the small package delivery business remains Deutsche Post World Net (DPWN). DHL, now under ownership of DPWN, generated $18.6 billion of that in 2003, with over 90% earned internationally, and an approximate 40% market share in international express, more than the United Parcel Service (UPS) and Federal Express (FedEx) combined. UPS is second with $33.5 billion in global revenues, but with only $5.6 billion earned from international package delivery. FedEx comes in at third with $24.7 billion total revenues, $4.6 billion generated outside the US. Despite being second best in international delivery, UPS and FedEx clearly dominate the world's largest domestic market, the more than $50 billion-a-year U.S. parcel delivery sector. With an estimated 70% of the market between them, they form a duopoly that has

a strong grip on the industry.

Perhaps more importantly, the express package industry may be revolutionizing the transportation industry, so far as the freight sector is concerned. Much of our understanding of the transportation industry has been based on a modal analysis. Morlok et al. [64] state that, "the traditional modal view of transportation... is out of date." This change of perspective has been largely driven by the expanding role of the package delivery industry over the last quarter century. Change in that period has been driven primarily by the private sector. There are two primary reasons as to why the package delivery industry has become so important in recent years. "One consists of changes in the way goods and services are produced and distributed in our economy-globalization, customized mass production, lean inventory management, rapid customer response, and growth in e-commerce, among others. The other is parcel service itself, which is at the vanguard of transportation service modernization with such features as differentiated time-definite service options, intermodal service, in-transit visibility, and data integration with the management systems of customers. Thus parcel service is a major element of the transportation infrastructure of the nation. It is essential for modern commerce."[64] The drive for innovation that FedEx and UPS apply to their parcel delivery division is rapidly spreading to the other services they provide, with improvements made that affect many areas of transportation and trucking.

FedEx is newer to the package delivery game, relative to its main rival, UPS. It was built on a set of innovative ideas for express delivery and this foundation has carried through to today. Over the last 40 years FedEx has become the company to turn to when a package has to "absolutely, positively" be there as quickly as possible. They are so synonymous with their service that the company name is used as a verb, as in to "FedEx" a package. A key to success at FedEx has been rapid adaptation to the needs of current and potential customers. They were the first company dedicated to overnight delivery, the first to offer Saturday delivery, the first to offer next day delivery by 10:30 AM, and the first to start Internet package status tracking services in 1994.

This tradition of innovation can be seen in each branch that is part of the FedEx family, including the branch that provides logistics solutions, FedEx Supply Chain Services. FedEx

acquired Caliber System in 1998, with both Caliber Logistics and Caliber Technology as part of the acquisition. The two divisions were merged to create FedEx Global Logistics and, in 2001, this group was realigned under FedEx Supply Chain Services (SCS), which is a part of the broader FedEx Services. FedEx SCS provides transportation and logistics management, consulting services and other customized supply chain solutions so customers can maximize potential and minimize costs. FedEx Services is responsible for the data management and networking expertise behind the package tracking capabilities used by FedEx Express and FedEx Ground, along with the e-commerce services that today's global marketplace demands.

FedEx SCS utilizes the fleets of various FedEx branches, such as Ground and Custom Critical, as well as the fleets of smaller contract carriers, to provide transportation services for a variety of customers with a wide range of needs. Multiple fleets are used for several reasons. Certain fleets may operate in a specific geographic region, provide a specific type of service, or simply provide needed capacity. This results in a customer receiving service from several transportation providers, many of whom are not directly communicating. In coordinating these services, FedEx SCS found that many vehicles were traveling with excess capacity. A primary problem was the lack of communication between the different fleets in use, such that two vehicles may be operating at half capacity along nearly identical routes. It was evident that a more efficient routing procedure was needed, and the possibility of using split loads was introduced. Also, FedEx believed that improving information sharing within the organization could result in significant cost benefits. However, implementing any of these improvements can be difficult, whether due to resistance to change within the corporation, difficulty in altering company operations, or mistrust of customers. FedEx SCS saw the need to quantify the value of split loads and information sharing in order to reveal inefficiencies in the transportation policy and provide an incentive for change. The heuristic presented in the previous chapter provided a good basis to begin testing these ideas.

## 5.2 Problem Description

In order to test the suggested changes, FedEx wished to analyze the implementation of services for one of their largest customers. FedEx currently services this customer with multiple fleets. In this regard, data was provided on all of the shipments requiring delivery on a daily basis for this customer. This data covered one week of operations. The shipment information is not expected to change from week to week over a six month time window, so this data is representative of the operations for a significant length of time. In addition to information on each service request, FedEx also provided the routes that were in use for fulfilling those requests.

With this information available, two questions could now be addressed:

1. What is the benefit of pooling all of the delivery requests that are being serviced individually and servicing them with fleets of vehicles that freely communicate?

2. What is the benefit of allowing split loads?

The first question is addressed by applying the heuristic without split loads to the data provided, and comparing the results with the routes operated by FedEx. The heuristic analyzes all loads in one pool, while the FedEx routes are developed by considering each load individually. That is, a load is placed on a route when the request for delivery occurs, and only some routing improvement is performed. Because this analysis is done with a heuristic, it does not provide the true benefit of communication between fleets. However, it does give a good representation of the scale of cost savings that may be found. Applying the heuristic with split loads and comparing the results with those from running the heuristic without split loads will address the second question.

In order to apply the heuristic to the FedEx problem, several modifications were made to fit the business rules under which FedEx SCS operates. The rules are as follows:

**1. No common depot:** There is no common depot for the vehicles used by FedEx SCS. In fact, the vehicles do not begin from or return to a depot on any of the routes. The first origin on a route is considered to be the initial node and the last destination is the final node. The depot was eliminated from the heuristic.

**2. One-way trip costs:** A different rate is charged dependent on whether a route is considered to be roundtrip or one-way. As is common within the trucking industry, roundtrips are highly encouraged as the vehicle and driver return close to the initial origin (which is viewed as their "home" base) at the end of a route. Based on FedEx guidelines, if the vehicle is within 20 miles of the initial origin at its final destination, the route is a roundtrip and a discounted rate of $1.00 per mile is used. Otherwise, a rate of $1.60 per mile is charged.

**3. Minimum route cost:** Each route has a minimum cost of $160. In this way, the fixed cost of operating a route is quantified, regardless of length. As the heuristic operated based on distances, this cost can be translated into a corresponding distance. Using the discounted rate of $1.00 per mile, this results in a minimum route distance of roughly 160 miles, while the rate of $1.60 results in a 100 mile minimum. Therefore, any route that was shorter than the minimum distance was counted as the corresponding minimum distance towards the total cost.

**4. Cost per stop:** FedEx operates with an average per stop cost of approximately $25. This accounts for the various costs associated with the vehicle loading or unloading at a facility. Again, as the heuristic operated based on distances, this was converted into a cost of 25 miles.

**5. Driver time per stop:** FedEx assumes that the driver will spend 60 minutes at a pickup location and 30 minutes at a dropoff. It can be expected that a driver can travel approximately one mile per minute when traveling, so these times were converted for the heuristic by adding 60 or 30 miles to the total cost per pickup or dropoff, respectively. Also, a location at which both a pickup and dropoff are made is treated as only a pickup and 60 miles are added to the total cost.

A route length maximum of 500 miles was used, as discussed in Chapter 3. The rule changes were made to the heuristic, and the data provided by FedEx was analyzed using the following experimental design.

## 5.3 Experimental Design

The data provided by FedEx SCS included information on 1182 lanes along which loads were moved at least once a week. The information pertinent to this study included:

1. address, including zip code, of each origin and destination,
2. size of load given as percentage of a truckload (assuming that all vehicle capacities are the same),
3. definition of each stop as origin or destination,
4. indication of when a load has returnables that must be delivered to the origin,
5. days during the week that the shipment must be transported, and how many times on that day.

A returnable is an item such as a packing crate that must be returned from a destination to the initial pickup location. Table 6 depicts several routes of data provided by FedEx. The data is organized such that the routes are presented exactly as FedEx operated them.

The name and address of the customers have been removed for privacy concerns. The fields in Table 6 are, from left to right, the alphanumeric route number generated by FedEx, the number of requests per day for each stop, the sequence the stops were visited on the route (99 indicates that the stop is for a returnable, and is always the same location as stop 0), the stop type (P is pickup, D is delivery and R is returnable), the zip code, the percentage of a truckload to be picked up from the stop, and the size of the returnable by percentage of a truckload. Note that the data does not indicate how much of a truckload is to be dropped off at a destination. In those situations where multiple destinations were serviced on a route, as in route BCC026, the loads were divided equally amongst the destinations. FedEx indicated that this is what generally occurs, and is therefore an appropriate approximation.

Routes with returnables were treated as two separate requests, one for the outbound pickup and delivery, and one for the returnable. Any returnable that must be delivered is given the delivery location as the origin and the original pickup location as the destination. For example, route BCC001 in Table 6 has a returnable that is 0.115 truckloads. This route

**Table 6:** Sample FedEx data set indicating all information provided for the loads delivered within one week.

| Route Number | Stop Freq SU | Stop Freq MO | Stop Freq TU | Stop Freq WE | Stop Freq TH | Stop Freq FR | Stop Freq SA | Stop Seq | Stop Type | Postal Code | Shipment Linear % | Returnables Shipment Linear % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BCC001 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | P | 48223 | 0.23 | 0.115 |
| BCC001 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | D | 48708 | 0 | 0 |
| BCC001 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 99 | R | 48223 | 0 | 0 |
| BCC002 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | P | 48601 | 0.201 | 0.201 |
| BCC002 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | P | 48071 | 0.268 | 0.268 |
| BCC002 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | D | 48708 | 0 | 0 |
| BCC002 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 99 | R | 48601 | 0 | 0 |
| BCC004 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | P | 49441 | 0.469 | 0.234 |
| BCC004 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | D | 48708 | 0 | 0 |
| BCC004 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 99 | R | 49441 | 0 | 0 |
| BCC007 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | P | 48706 | 0.362 | 0 |
| BCC007 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | D | 48708 | 0 | 0 |
| BCC007 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | D | 48711 | 0 | 0 |
| BCC008 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | P | 48706 | 0.362 | 0 |
| BCC008 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | D | 48708 | 0 | 0 |
| BCC008 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | D | 48711 | 0 | 0 |
| BCC026 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | P | 15834 | 0.58 | 0.29 |
| BCC026 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | D | 48708 | 0 | 0 |
| BCC026 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | D | 48601 | 0 | 0 |
| BCC026 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 99 | R | 15834 | 0 | 0 |

becomes a request to deliver 0.23 truckloads from zip code 48223 to 48708 and a request to deliver 0.115 truckloads from zip code 48708 to 48223. The delivery of a returnable prior to the delivery of its corresponding load is allowed. This is because all deliveries are made at least once a week, and most are made every day. Therefore, a returnable that is picked up before the delivery of the original load could be the returnable that was left on the last delivery, the day before or earlier in the week. While a returnable would ideally not remain at a destination for an extended period of time, FedEx allowed for this occurrence.

Each address was geocoded using the corresponding zip code. Geocoding involves the assignment of $x$ and $y$ coordinates to a location, based on some geographical identifier, which in this case was the zip code. There were no pickups with deliveries in the same zip code. However, as some pickups at neighboring locations shared the same zip code, an additional two letter identifier taken from the customer name was added to each zip code to distinguish the individual stops. Euclidean distances were used between the geocoded locations. In most instances, the distances were long enough that the value of the Euclidean distances was very close to the real distances, and in some cases, slightly more accurate than the distances used by FedEx.

The heuristic run time for the data from one full day was in excess of 20 hours for the busiest day (Monday). This day required the delivery of over 340 truckloads. Because of this, the data for each day was divided to reduce the overall run time. The full data sets for each day were also run for comparison. The load requests were grouped together based on the zip code of the origin location, such that transportation requests originating from neighboring regions were in the same set. The divisions were also partially based on the total load size to be delivered, as well as the total number of origins and destinations. This was to prevent one data set from requiring a significantly greater computational time than any other set. The data for Monday and Tuesday was partitioned into 5 sets, while Wednesday, Thursday and Friday were partitioned into 4 sets. The data for the former two days was more naturally partitioned into 5 sets instead of 4 based on the specifications indicated. The data for Saturday and Sunday was not partitioned as the number of loads requiring service was significantly lower for these days. All experiments were run on a 2.4

GHz Xeon processor with a 400Mhz frontside bus and 2 GB RAM.

The cost of the routes as they were run by FedEx was determined using Microsoft Excel. All of the same rules were applied (cost per stop, penalty for one-way trip, etc.) and the same coordinates were used for each location.

## *5.4 Experimental Results*

Table 7 presents the results for the FedEx data partitioned as described above. Each partition was run under two scenarios, with and without split loads. Output from the heuristic is provided on the number of iterations for which the heuristic performs the local improvement algorithms, the cost of the routes in the partition, the number of routes used and the computational time required, in minutes. The number of iterations for the instances without split loads was fixed at three, as no improvement was seen after this many iterations of local improvements. Also, the size of the partition is represented by the number of origins and destinations requiring service, and the total number of truckloads to be delivered. These matrices were sparse, as an average of only 6% of origin-destination pairs had a request for service. The final value provided is the percent increase in cost when split loads are not allowed. The difference between the cost with split loads and the cost without is not significant, particularly when considering the additional computational time required when using split loads. Section 5.5 evaluates the FedEx business rules and data, and determines the impact these have on the significant decrease in the benefit of split loads. Despite the lack of cost benefit, there was a reduction in the number of routes used, which, if given a minimum fixed cost per route, can be of importance.

Table 8 presents a comparison between the costs under the current FedEx operation and those found by the heuristic, both with and without split loads. The results are organized such that the first main column of data contains results on the sum total of the partitioned sets presented in Table 7, the second column contains the results from running the full data set for each day, and the third column presents the FedEx costs and a comparison with those costs. The same information is provided as in the previous table for each of the problem instances. Again, the difference between the cost with and without split loads is minimal,

**Table 7:** Heuristic results, partition dimensions and average percentage cost increase without split loads for partitioned data.

| | | 1 | | 2 | | 3 | | 4 | | 5 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Split | No Split | Split | No Split | Split | No Split | Split | No Split | Split | No Split |
| **M O N** | Iterations | 997 | 3 | 1017 | 3 | 773 | 3 | 1115 | 3 | 800 | 3 |
| | Cost | 60235 | 61016 | 27516 | 28009 | 28503 | 28700 | 22363 | 22986 | 32093 | 32812 |
| | Routes | 86 | 91 | 43 | 47 | 36 | 39 | 33 | 39 | 51 | 54 |
| | Time (min.) | 42 | < 0.1 | 7 | < 0.1 | 20 | < 0.1 | 15 | < 0.1 | 12 | < 0.1 |
| | # Orig. X # Dest. | 24 X 57 | | 40 X 33 | | 16 X 44 | | 21 X 44 | | 44 X 36 | |
| | Truckloads | 97.55 | | 44.47 | | 81.49 | | 57.03 | | 62.10 | |
| | % Cost Increase | 1.30 | | 1.79 | | 0.69 | | 2.79 | | 2.24 | |
| **T U E** | Iterations | 976 | 3 | 978 | 3 | 990 | 3 | 1085 | 3 | 676 | 3 |
| | Cost | 59001 | 59664 | 27748 | 28257 | 32301 | 32664 | 22996 | 24111 | 20461 | 20481 |
| | Routes | 81 | 84 | 39 | 44 | 45 | 47 | 33 | 42 | 37 | 39 |
| | Time (min.) | 40 | < 0.1 | 9 | < 0.1 | 24 | < 0.1 | 14 | < 0.1 | 3 | < 0.1 |
| | # Orig. X # Dest. | 25 X 54 | | 46 X 37 | | 19 X 43 | | 18 X 45 | | 33 X 20 | |
| | Truckloads | 94.77 | | 45.29 | | 96.44 | | 60.03 | | 34.69 | |
| | % Cost Increase | 1.12 | | 1.83 | | 1.12 | | 4.85 | | 0.10 | |
| **W E D** | Iterations | 861 | 3 | 952 | 3 | 922 | 3 | 1157 | 3 | | |
| | Cost | 58525 | 59035 | 29672 | 29850 | 35998 | 36430 | 39429 | 40484 | | |
| | Routes | 85 | 87 | 46 | 48 | 49 | 55 | 58 | 61 | | |
| | Time (min.) | 36 | < 0.1 | 10 | < 0.1 | 26 | < 0.1 | 35 | < 0.1 | | |
| | # Orig. X # Dest. | 24 X 55 | | 47 X 37 | | 24 X 50 | | 46 X 48 | | | |
| | Truckloads | 94.85 | | 49.85 | | 101.68 | | 90.21 | | | |
| | % Cost Increase | 0.87 | | 0.60 | | 1.20 | | 2.68 | | | |
| **T H U R** | Iterations | 1146 | 3 | 970 | 3 | 1221 | 3 | 706 | 3 | | |
| | Cost | 55619 | 56358 | 39673 | 40297 | 32887 | 34029 | 29302 | 29439 | | |
| | Routes | 78 | 82 | 55 | 61 | 44 | 52 | 49 | 50 | | |
| | Time (min.) | 36 | < 0.1 | 25 | < 0.1 | 37 | < 0.1 | 10 | < 0.1 | | |
| | # Orig. X # Dest. | 25 X 55 | | 46 X 41 | | 28 X 55 | | 41 X 33 | | | |
| | Truckloads | 92.00 | | 83.93 | | 93.15 | | 59.18 | | | |
| | % Cost Increase | 1.33 | | 1.57 | | 3.47 | | 0.47 | | | |
| **F R I** | Iterations | 995 | 3 | 984 | 3 | 996 | 3 | 767 | 3 | | |
| | Cost | 58025 | 58621 | 40959 | 41597 | 31337 | 32047 | 33633 | 33780 | | |
| | Routes | 81 | 86 | 58 | 62 | 45 | 48 | 53 | 54 | | |
| | Time (min.) | 39 | < 0.1 | 25 | < 0.1 | 27 | < 0.1 | 22 | < 0.1 | | |
| | # Orig. X # Dest. | 25 X 55 | | 47 X 43 | | 26 X 57 | | 44 X 34 | | | |
| | Truckloads | 92.22 | | 86.74 | | 80.81 | | 69.28 | | | |
| | % Cost Increase | 1.03 | | 1.56 | | 2.26 | | 0.44 | | | |
| **S A T** | Iterations | 826 | 3 | | | | | | | | |
| | Cost | 12783 | 12953 | | | | | | | | |
| | Routes | 18 | 22 | | | | | | | | |
| | Time (min.) | 3 | < 0.1 | | | | | | | | |
| | # Orig. X # Dest. | 24 X 22 | | | | | | | | | |
| | Truckloads | 31.78 | | | | | | | | | |
| | % Cost Increase | 1.33 | | | | | | | | | |
| **S U N** | Iterations | 764 | 3 | | | | | | | | |
| | Cost | 8513 | 8573 | | | | | | | | |
| | Routes | 10 | 12 | | | | | | | | |
| | Time (min.) | 1 | < 0.1 | | | | | | | | |
| | # Orig. X # Dest. | 14 X 12 | | | | | | | | | |
| | Truckloads | 18.27 | | | | | | | | | |
| | % Cost Increase | 0.71 | | | | | | | | | |

**Table 8:** Heuristic results, data dimensions, comparison with FedEx costs, and average percentage cost increase without split loads for full and partitioned data.

| | | Partitions | | Total | | FedEx | | |
|---|---|---|---|---|---|---|---|---|
| | | Split | No Split | Split | No Split | | % Cost Diff. | |
| **M O N** | Iterations | N/A | N/A | 1139 | 3 | N/A | | |
| | Cost | 170709 | 173522 | 157850 | 158817 | 191212 | 12.01 | Partitions - Split |
| | Routes | 249 | 270 | 207 | 216 | 359 | 10.19 | Partitions - No Split |
| | Time (min.) | 95 | < 0.2 | 1203 | 3 | N/A | 21.14 | Total - Split |
| | % Cost Increase | 1.65 | | 0.61 | | | 20.40 | Total - No Split |
| | Orig. X Dest. | 145 X 123 | | | | | | |
| | Truckloads | 342.63 | | | | | | |
| **T U E** | Iterations | N/A | N/A | 1140 | 3 | N/A | | |
| | Cost | 162507 | 165177 | 151525 | 152588 | 185563 | 14.19 | Partitions - Split |
| | Routes | 235 | 256 | 199 | 211 | 350 | 12.34 | Partitions - No Split |
| | Time (min.) | 90 | < 0.2 | 832 | 3 | N/A | 22.46 | Total - Split |
| | % Cost Increase | 1.64 | | 0.70 | | | 21.61 | Total - No Split |
| | Orig. X Dest. | 141 X 117 | | | | | | |
| | Truckloads | 331.22 | | | | | | |
| **W E D** | Iterations | N/A | N/A | 1253 | 3 | N/A | | |
| | Cost | 163625 | 165799 | 153493 | 155079 | 186730 | 14.12 | Partitions - Split |
| | Routes | 238 | 251 | 201 | 211 | 353 | 12.62 | Partitions - No Split |
| | Time (min.) | 107 | < 0.2 | 913 | 3 | N/A | 21.65 | Total - Split |
| | % Cost Increase | 1.33 | | 1.03 | | | 20.41 | Total - No Split |
| | Orig. X Dest. | 141 X 119 | | | | | | |
| | Truckloads | 336.596 | | | | | | |
| **T H U R** | Iterations | N/A | N/A | 946 | 3 | N/A | | |
| | Cost | 157481 | 160123 | 147582 | 148735 | 181895 | 15.50 | Partitions - Split |
| | Routes | 226 | 245 | 199 | 205 | 346 | 13.60 | Partitions - No Split |
| | Time (min.) | 108 | < 0.2 | 727 | 3 | N/A | 23.25 | Total - Split |
| | % Cost Increase | 1.68 | | 0.78 | | | 22.30 | Total - No Split |
| | Orig. X Dest. | 140 X 115 | | | | | | |
| | Truckloads | 328.26 | | | | | | |
| **F R I** | Iterations | N/A | N/A | 1367 | 3 | N/A | | |
| | Cost | 163903 | 165847 | 153586 | 154949 | 182763 | 11.51 | Partitions - Split |
| | Routes | 236 | 248 | 202 | 215 | 345 | 10.20 | Partitions - No Split |
| | Time (min.) | 114 | < 0.2 | 956 | 3 | N/A | 19.00 | Total - Split |
| | % Cost Increase | 1.19 | | 0.89 | | | 17.95 | Total - No Split |
| | Orig. X Dest. | 142 X 117 | | | | | | |
| | Truckloads | 329.06 | | | | | | |
| **S A T** | Iterations | 826 | 3 | 826 | 3 | N/A | | |
| | Cost | 12783 | 12953 | 12783 | 12953 | 17732 | 38.71 | Partitions - Split |
| | Routes | 18 | 22 | 18 | 22 | 42 | 36.90 | Partitions - No Split |
| | Time (min.) | 3 | < 0.1 | 3 | < 0.1 | N/A | 38.71 | Total - Split |
| | % Cost Increase | 1.33 | | 1.33 | | | 36.90 | Total - No Split |
| | Orig. X Dest. | 24 X 22 | | | | | | |
| | Truckloads | 31.78 | | | | | | |
| **S U N** | Iterations | 764 | 3 | 764 | 3 | N/A | | |
| | Cost | 8513 | 8573 | 8513 | 8573 | 11233 | 31.95 | Partitions - Split |
| | Routes | 10 | 12 | 10 | 12 | 25 | 31.02 | Partitions - No Split |
| | Time (min.) | 1 | < 0.1 | 1 | < 0.1 | N/A | 31.95 | Total - Split |
| | % Cost Increase | 0.71 | | 0.71 | | | 31.02 | Total - No Split |
| | Orig. X Dest. | 14 X 12 | | | | | | |
| | Truckloads | 18.27 | | | | | | |

both for the cumulation of the partitioned data and for the full data set. The full data set has less benefit from split loads than the partitioned data because there are relatively fewer split loads tested for the larger set. The heuristic will generate a similar number of split loads for both a partition and a full data set if there is no cost reduction found for the loads tested. However, as the range of split loads generated is a more thorough representation for a smaller data set, there is more likelihood that one of the split loads tested will result in a cost reduction. Despite a lack in benefit from generating split loads, the overall cost of the full data set is almost 10% less than that of the cumulative partitions. Testing the full data set provides more opportunities for local improvements to be made as there are many more route combinations available and origins and destinations that can be swapped or inserted. Circumstances may dictate whether the reduction in computational time from using the partitioned data outweighs the gain in overall cost.

The column with the costs of the current FedEx operation also includes a comparison with the four heuristic costs generated: the cost for the cumulative partitioned data with and without split loads and the cost of the full data set with and without split loads. Each of these percentages indicates a significant improvement in the use of the heuristic over the routes that were selected by FedEx. Pooling the loads and rerouting the service requests results in a marked cost savings. Even when the heuristic was applied to the partitioned data, in which the loads were initially pooled and then broken into several sets, a reduction in cost was found.

## *5.5 Evaluation of FedEx Rules and Data*

The results when applying the heuristic to the FedEx data countered those results found when using the randomly generated data in Chapter 3 in that there was little benefit to allowing split loads. This is a factor of both the characteristics of the FedEx data and the additional rules (cost per stop, penalty for one-way trip, etc.) that were implemented for the FedEx case. We initially analyze the effect that the rules have on the benefit of split loads.

In order to make the effect of the rules most visible, the heuristic was tested on data with

loads that are most likely to be split. As shown in Chapter 3, these are loads in the range of 0.51 - 0.6 truckloads. The data used for this test is the same as that used in Chapter 3, and it consists of the 15 different load and location combinations with 75 service requests. The heuristic was initially applied to the data with adherence to all of the FedEx rules. The heuristic was then reapplied with the rules individually eliminated. The following rule changes were tested:

1. elimination of per stop costs, including the cost of 25 miles for any stop, and the costs of 60 and 30 miles for origins and destinations, respectively;
2. elimination of the penalty for a one-way trip in which the vehicle does not return close to its original starting location at the end of a route;
3. addition of a central depot from which all routes begin and end;
4. elimination of a minimum route length.

After initial testing showed that the per stop cost played a pivotal role in the reduction of the benefit of split loads, two additional rule changes were tested:

5. multiplying the per stop costs by 1/2, so the cost for any stop is 12, and the costs for origins and destinations are 30 and 15, respectively
6. multiplying the per stop costs by 1/5, so the cost for any stop is 5, and the costs for origins and destinations are 12 and 6, respectively.

Table 5.5 presents the average percentage increase in cost from allowing split loads to prohibiting them for the 15 data sets, under each of the different rule conditions. The first value indicates the percentage increase with all of the FedEx rules applied to the heuristic, the second is without any rules, as was tested in Chapter 3, the third is without any per stop costs, the fourth is with those costs multiplied by 1/2, the fifth is with those costs multiplied by 1/5, the sixth is without a penalty for one-way trips, the seventh is without a minimum route length, and the eighth is with a central depot added.

As these values indicate, the per stop costs are a primary reason for the reduction in the benefit found for using split loads. The per stop costs accounted for the most significant

**Table 9:** Benefit of split loads for loads in range 0.51-0.6 with FedEx rules and with each rule removed or altered.

| | All FedEx rules | No rules | All FedEx rules with: | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | no per stop cost | 1/2 per stop cost | 1/5 per stop cost | no one-way trip penalty | no minimum length | depot |
| % Cost increase | 22.24 | 40.61 | 42.88 | 25.80 | 30.71 | 21.98 | 19.01 | 20.52 |

portion of the cost, with 68% of the total cost with split loads and 65% without split loads. However, the benefit from eliminating stops was much smaller than reducing routing costs. While the actual costs of the routes increased by 32% without split loads, the costs of the stops increased by only 18%. More stops were visited without split loads; yet, most impact from the use of split loads was felt with the reduction in routing costs. This result is not surprising, as generating a split load generally forces a route to visit an additional origin and destination. Even with the costs of a stop reduced by 1/2, the benefit is marginally increased. Further reducing the costs per stop does improve the benefit, as shown when the costs are multiplied by 1/5, but they must be completely eliminated to realize the most savings when using split loads. This result is significant, as most real routing situations have at least some cost per stop. However, even with the per stop costs, there is a benefit of over 20% when the loads are within the range 0.51 - 0.6. While this is a rather specific window for loads to fall within, it does show that there may be a cost benefit for using split loads with a cost per stop.

Additionally, this test indicated that the rules regarding a penalty for one-way trips, a minimum route length and the elimination of the depot improved the benefit of split loads. Splitting loads creates the opportunity for more roundtrip routes. An origin or destination may be visited multiple times when loads into or out of that location are split, such that the stop may be used to complete a roundtrip on several routes. Thus, the use of split loads may reduce the occurrence of the penalty applied for one-way trips, and therefore, reduce the overall cost. Similarly, the elimination of the depot slightly improves the benefit of split loads. When the vehicle must return to the depot, there can be no penalty for one-way

trips. As mentioned, eliminating this penalty reduces the benefit of split loads. Also, each route both with and without split loads has the additional cost of the vehicle leaving from and returning to the depot. This cost increases the overall cost such that the relative benefit of split loads is reduced. The use of a minimum route length raises the benefit of split loads as it increases the cost of routing without split loads. If there is no minimum route length, each route without split loads consists of an arc between an origin and a destination, with no need to combine routes for cost savings. With a minimum route length, these routes are combined at some cost in order to generate routes that are closer to the minimum. The additional cost of either combining the routes or using the minimum route length results in a more significant cost increase from routing with split loads.

Given the impact of these rules, the costs associated with each stop were negated from the heuristic, and it was reapplied to the FedEx data. The increase in the benefit of split loads without these costs averaged just below 1% for each day, an almost negligible change. This indicates that in this situation, most of the benefit is lost due to the characteristics of the data, particularly the range of load sizes and the sparsity of the service request matrix. A significant number of the load sizes were lower than 0.5 truckloads. There were quite a few requests for service for more than one truckload. However, the fractional portion of those requests was generally below 0.5 truckloads, such as a request for 2.3 truckloads to be serviced. This is important because of the way that routes are initially generated. A separate route is generated for each full truckload, with an additional route for any remaining fraction of a truckload. A full truckload is less likely to be split than one with just over 0.5 truckloads, as is a load sized below 0.5 truckloads. Almost 60% of the loads were below 0.5 truckloads, or had a fractional portion that was below 0.5 truckloads. Therefore, the load sizes of the FedEx data are not extremely conducive to splitting.

To test the effect on the benefit of split loads, the load sizes were modified for each of the partitioned data sets. The load size for each service request was changed to a randomly generated value in the range 0.51 - 0.6 or 1.51 - 1.6. The origins and destinations were not altered. The range within which the load size was selected was determined such that the total load size for a data set was comparable to the original total. The heuristic was

run on the new data with all of the FedEx rules applied. Table 5.5 presents the average percentage cost increase without split loads for each weekday (Saturday and Sunday were not included as the results for these days were skewed due to the small size of the service request matrix), as well as the average over the 5 weekdays. The benefit of using split loads

**Table 10:** Benefit of split loads with FedEx rules when FedEx load sizes are altered.

| Load range | Monday | Tuesday | Wednesday | Thursday | Friday | Average |
|---|---|---|---|---|---|---|
| 1.51 - 1.6 / 0.51 - 0.6 | 7.41 | 7.50 | 8.18 | 8.12 | 7.31 | 7.58 |
| 0.51 - 0.6 | 14.67 | 14.47 | 14.74 | 14.27 | 13.79 | 14.03 |

becomes much more evident with the altered load sizes. However, it is still well below the 22% benefit found when using the randomly generated instances with loads in the range 0.51 - 0.6. As stated before, full truckloads are not as likely to be split and the use of loads in the range 1.51 - 1.6 has an impact on the benefit.

To determine what effect the full truckloads may have, the load sizes were again altered. Each service request was changed to a randomly generated value in the range 0.51 - 0.6. With the lack of any full truckloads, the total load size for each data set was on average almost 30% lower then the original total. The average percentage cost increases for the loads in this range are also presented in Table 5.5. The benefit is almost doubled through the elimination of the full truckloads. However, it is still more than 7% below the benefit found with the randomly generated data sets used in Chapter 3. This difference can be explained by the sparseness of the service request matrix. In the randomly generated problem instances, every origin-destination combination has a load associated with it. Therefore, when a load is split, there is a good chance that load can be placed on another route segment where the origin or destination is shared with another load. This results in a much lower cost for splitting a load as only one other origin or destination must be visited, and in some cases no additional stops must be made. This is not true of the FedEx data. While many origins have loads requiring transportation to multiple destinations (and many destinations receive loads from multiple origins), the frequency with which this occurs is not as regular. Fewer than 37% of origins send loads to more than one destination, with only 12 % sending

loads to more than three destinations. The values are similar for destinations, with less than 38% of destinations receiving loads from more than one origin and only 16% receiving loads from more than three origins. Therefore, when a split load is created, most instances result in a route visiting an additional origin and destination. There is still a benefit for using split loads, but it is not as significant.

## *5.6 Conclusions*

In this chapter, a case study involving Federal Express was presented. FedEx was finding significant excess capacity in the various fleets it was using to service its clients. Pooling the service requests and using split loads are two options that were considered. Several business rules used by FedEx were applied to the heuristic presented in Chapter 3, and a data set representative of weekly operations for a major client was tested using the adapted heuristic. While pooling the loads was found to have significant benefit in reducing the overall cost of servicing this client, allowing split loads resulted in almost no cost reduction. The various FedEx rules were tested to determine the effect that each may have on the benefit of using split loads, and the addition of a cost for each stop was found to have a marked impact when evaluating randomly generated data sets. However, the FedEx data did not respond similarly to the elimination of this rule. Additional testing found the load sizes in the data were not conducive to splitting, with most loads below 0.5 truckloads or at full truckload size. Furthermore, the limited number of loads that share an origin or destination in the FedEx data set restricted the benefit that could be found with split loads.

# CHAPTER VI

# PDPSL WITH ORIGIN/DESTINATION CONSTRAINT

This chapter introduces the Pickup and Delivery Problem with Split Loads with an additional constraint that all origins must be visited before any destinations on each route (CPDPSL). As was shown in Chapter 3, the exact solution of the general PDPSL problem is difficult for even the smallest problems. Adding the constraint of visiting origins prior to any destinations on a route reduces the state and action sets, allowing for the solution of larger problems to optimality, while also providing an upper bound on the solution of the general PDPSL. Furthermore, the vehicle is required to return to the depot whenever it is emptied. With these additional constraints, the PDPSL becomes very similar to the Split Delivery Vehicle Routing Problem (SDVRP). Most structural characteristics of the SDVRP can be generalized for the CPDPSL. Section 6.1 provides the formulation for the revised problem with the additional constraints. Section 6.2 presents several important properties of the CPDPSL that are generalized from the SDVRP. Section 6.3 explains how the action space may be reduced, while Section 6.4 discusses the use of the reduced action space in creating a finite state space. The A* algorithm used for the solution of the CPDPSL is described in Section 6.5. Sections 6.6 and 6.7 present the design and results of the experiment used for analysis, and Section 6.8 provides a conclusion for the chapter.

## *6.1 Model Formulation*

Let the graph $G = (N, E)$ serve as a model of a fully connected road network, where $N$ is the finite set of nodes, modeling intersections, and $E \subseteq N \times N$ is the set of directed arcs modeling one-way roads between intersections. That is, $(n, n') \in E$ if and only if there is a road that permits traffic to flow from intersection $n$ to intersection $n'$. Let $SCS(n) = \{n' : (n, n') \in E\}$ be the successor set of node $n$. We assume throughout that $n \notin SCS(n)$; thus, idling at an intersection is not permitted.

Pickups can be made at the origin nodes, $O \subset N$, and deliveries can be made at the destination nodes, $D \subset N$. We remark that $O \cap D$ may not be null; that is, there may be nodes that serve as both origin and destination nodes (e.g., a crossdock). However, if a node falls in both sets, it is treated as both an origin and destination. There also exists a node $Depot \subset N$, where $Depot \notin O \cap D$. The vehicle leaves from, and returns to, the $Depot$ empty at both the beginning and the end of a route. Further, the vehicle must return to the depot whenever it is empty. Because the vehicle starts from and returns to the depot, each route may be serviced by a separate vehicle. However, for simplicity of analysis, we assume that there is a single pickup and delivery vehicle. We assume $O \cup D \cup \{Depot\} = N$. We now make the additional assumption that all origins on a route are visited prior to any destination. A route is a sequence of nodes $(Depot, n_1, ..., n_Q, Depot)$, with $n_q \in N$ for all $q = 1, ..., Q$, and $n_{q+1} \in SCS(n_q)$ for all $q = 1, ..., Q-1$. Visiting all origins on a route first requires that $q < q'$ for all $n_q \in O$ and $n_{q'} \in D$.

Let $a(i,j)$ be the amount of goods stored at $i \in O$ destined for $j \in D$, and let $a = \{a(i,j) : (i,j) \in O \times D\}$. We call $a$ the inventory position, where it is said to be feasible ($a \in \mathcal{F}$) if and only if $a(i,j) \geq 0$ for all $(i,j) \in O \times D$. Also, without loss of generality we assume that the capacity of the vehicle is 1.

A decision epoch occurs each time the vehicle arrives at the $Depot$. Let $t_q$ be the time of the $q$th decision epoch. Let $T$ be a finite integer indicating that action selection terminates at a time no later than $T-1$, and $Q$ is the final decision epoch. $T$ may represent the maximum length of a route. Just prior to the $q$th decision, assume that $a_q(i,j)$ is the amount of goods at $i \in O$ that is destined for $j \in D$, where $a_q = \{a_q(i,j) : (i,j) \in O \times D\}$.

An action for this DP is a set of origins and destinations to be visited on a route and the number of truckloads to be transported between these locations by a single (capacitated) truck. We represent such an action with a non-negative vector $x = \{x(1,1), ..., x(|O|, |D|)\}$, where $|O|$ and $|D|$ are the cardinalities of sets $O$ and $D$ and, in order to ensure feasibility, $x(i,j) \leq a(i,j)$ for all $i$ and $j$. Furthermore, because a route visits all origins prior to any destination, the vehicle can not pick up more than the constraints of vehicle capacity allow on any given route. Therefore, $\sum_{i \in O, j \in D} x(i,j) \leq 1$. If $a_q$ is the inventory position at time

$t_q$ and $x_q$ is the selected action, then the inventory position at $t_q$ becomes $f_1(a_q, x_q)$, where:

$$[f_1(a_q, x_q)](i, j) = a_q(i, j) - x_q(i, j)$$

Let the action set $A(a, t)$ represent the set of all actions available to the driver at time $t$, given the state of loads to be delivered is $a$.

A decision rule at time $t$ is a function $\delta(\cdot, t) : x \to A(\cdot, t)$ that selects an available action. Thus, $\delta(a, t) \in A(a, t)$. A policy is a sequence of decision rules $\pi = \{\delta(\cdot, 1), \delta(\cdot, 2), ..., \delta(\cdot, T-1)\}$ that selects actions based on the current inventory position. We remark that $\delta(\cdot, t)$ is implemented only if $t$ is a decision epoch.

As indicated previously, every route originates at some depot, visits one or several origins, continues to one or several destinations, and finally returns to the depot. Each route can be represented by a vector $r \in \{0, 1\}^{|O|\times|D|}$, where $r(i, j) = 1$ if and only if the truck following this route delivers a load from the $i$th origin to the $j$th destination. We define $x^r$ as the vector of loads to be picked up and delivered on route $r$, where $x^r$ must satisfy the inequality $x^r \leq r$. That is, if a load is delivered on a route, both the origin and destination must be visited on that route. Thus, if a truck is to service the load vector $x$, $r_x$ describes the origins and destinations that must be visited. We can now more specifically define the set of all possible load combinations that a truck can make given the inventory position $a$ at time $t$ as

$$A(a, t) = \{x \in \mathbb{R}_+^{|O|\times|D|} : x \leq a, \sum_{i\in O, j\in D} x(i, j) \leq 1, r_x \in R\} \tag{67}$$

where the collection of all feasible routes is denoted by $R = \{r^1, ..., r^K\}$.

The above specification of a route does not describe the order in which the origins or destinations are visited by the truck. We will, however, assume that they are visited in the order leading to the minimal cost of the route specified by the vector $r$ throughout this paper (i.e., the travelling salesman tour with the restriction that origins are visited prior to destinations). The total cost of executing route $r$ is then the sum of travel costs between the consecutive origins, from the final origin to the first destination, and between the consecutive destinations on the route (according to the minimal cost routing), and is

denoted by $c(r)$. The cost structure is comprised of the sum of the travel costs. Then, the problem criterion is

$$v^{\pi}(a,1) = \sum_{q=1}^{Q} c(x_q)$$

conditioned on the use of policy $\pi, a_1 = a$, and $t = 1$. We allow that the cost of routing is $c(x_q) = c(r_{x_q})$. The problem objective is to find a policy $\pi^*$, called an optimal policy, such that $v^{\pi^*}(a,1) \leq v^{\pi}(a,1)$ for all policies $\pi$ and all inventory positions $a$.

### 6.1.1 Optimality Equations

Let $v(a,t)$ be the optimal cost to be accrued from epoch $t$ through the remainder of the problem horizon, given that $a$ is the inventory position just before $t$. Then, based on the results in Puterman [79] and elsewhere, the optimality equation is as follows for $t < T$:

$$v(a,t) = \min_{x}\{c(x) + v(f_1(a,x), t')\}, \tag{68}$$

where the minimization is with respect to all $x \in A(a,t)$.

The solution of the optimality equation is unique, and $v(a,t) = \min_{\pi} v^{\pi}(a,t)$, for all $a$ and $t$. We refer to $v(a,t)$ as the expected cost or cost-to-go function. A necessary and sufficient condition for $\pi^*$ to be optimal is that for every $t \in \{1, ..., T\}$ and $a \in \mathcal{F}$, the action selected causes the minimum in the optimality equations to hold. The same monotonicity conditions hold for the CPDPSL as for the PDPSL described in Chapter 2, except $v(a,t)$ monotonically increases with $a$.

## *6.2 Properties of Optimal Collections of Routes*

The size and combination of loads that may be serviced in the CPDPSL is limited by the capacity of the vehicle. This restriction is similar to that found in the Split Delivery Vehicle Routing Problem (SDVRP). This similarity results in many shared structural characteristics. In this section, we generalize several of the theoretical results from the SDVRP found in Dror and Trudeau [21, 22] and Lee et al. [58].

Lemmas 6 and 8 generalize results by Dror and Trudeau [22] for the SDVRP. We show that the number of split loads allowed on a route is limited, restricting the different load

combinations possible for any route. We say that two routes, $r^1$ and $r^2$, have a split load in common if $r^1(i,j) = r^2(i,j) = 1$.

**Lemma 6** *There exists an optimal solution in which no two routes have more than one split load origin-destination pair in common.*

Proof: Let $R^*$ be an optimal collection of routes. Suppose that $r^1$ and $r^2$ each have two split loads being delivered from pickup locations $i$ and $k$ to destinations $j$ and $l$, respectively. The origin $i$ may equal $k$ and the destination $j$ may equal $l$. Let $x^{r^1}(i,j), x^{r^2}(i,j), x^{r^1}(k,l)$, and $x^{r^2}(k,l)$ be the corresponding loads for routes 1 and 2. Without loss of generality, suppose that $x^{r^1}(i,j) = \min\{x^{r^1}(i,j), x^{r^2}(i,j), x^{r^1}(k,l), x^{r^2}(k,l)\}$. Consider the alternative feasible routing decision and the corresponding loads:

$$\bar{x}^{r^1}(i,j) = 0, \bar{x}^{r^2}(i,j) = x^{r^2}(i,j) + x^{r^1}(i,j),$$
$$\bar{x}^{r^1}(k,l) = x^{r^1}(k,l) + x^{r^1}(i,j), \bar{x}^{r^2}(k,l) = x^{r^2}(k,l) - x^{r^1}(i,j)$$

Now, $r^1$ can omit the stops at $i$ and $j$ as that load is serviced by $r^2$. The new loads are feasible, and the new routing has one fewer load serviced. By the triangular inequality, this can be no more expensive than the original routing. Continuing inductively, we can construct an optimal collection of routes with no two routes transporting split loads from the same two origin-destination pairs. □

We now define a k-split cycle for the CPDPSL.

**Definition 7** *Given $k$ split loads for origin-destination pairs $p_1, p_2, ..., p_k$ and $k$ routes. Suppose that route 1 services the loads for the pairs $p_1$ and $p_2$, route 2 delivers the loads for $p_2$ and $p_3$, and so on. Finally, route $k$ services the loads for $p_k$ and $p_1$. This subset of origin-destination pairs $\{p_i\}_{i=1}^k$ is called a $k$-split cycle.*

The following lemma generalizes Lemma 6.

**Lemma 8** *There exists an optimal collection of routes in which there are no $k$-split cycles (for any $k \geq 2$).*

Proof: This proof follows a similar argument to that of the previous lemma. □

Restricting the number of split loads that can be shared between routes leads to a result that allows for a significant reduction in the action space available at each decision epoch.

We show that the combinations of loads that may be created for each route are limited such that only one split load may be placed on a route.

**Theorem 9** *There exists an optimal solution for which each route will have at most one split load.*

Proof: Let $R^*$ be an optimal collection of routes and $r^t$ be the $t$th route in $R^*$. Suppose that $r^1$ carries $n$ split loads, such that $n > 1$. The routes may be reordered such that $r^1$ is the last route traveled. The ordering of the routes is independent of cost as the vehicle begins and ends empty for each route. Therefore, changing the order in which $r^1$ is serviced does not affect optimality. If $r^1$ is the last route, all $n$ split loads on this route are no longer split as they complete the service of a load. Now, suppose $r^2$ also carries $n$ split loads, with $n > 1$. The ordering can be changed again such that $r^2$ is the second to last route. Because there are no $k$-split cycles, $r^1$ and $r^2$ share at most one origin-destination pair. Thus, the reordering will reduce the number of split loads on $r^2$ to at most one. This reordering can be repeated for each route with $n > 1$ split loads until all routes have at most one split load. □

## *6.3 Action Space Reduction*

The optimality equation presented in Section 6.1.1 is computationally undesirable due to the uncountably infinite state and action spaces. However, the properties of an optimal solution described above allow for the action space of the problem, $A(a, t)$, to be reduced to a finite set without loss of optimality. It has been shown that each route can carry at most one split load. Therefore, at every decision epoch, only those routes with load combinations that satisfy this property should be considered, leaving a finite action set.

The set of loads to be delivered on the next route can be represented by the pair $(E, I)$, where $E \subseteq \{O \times D\}$ is the set of origin/destination pairs fully serviced by route $r$ and $I$ is the set of origin/destination pairs with a split load serviced. A load can not be in both sets, such that $E \cap I = \emptyset$, and the cardinality of I is at most one. Given $(E, I)$, it is possible

to generate the route $r$ taken by the vehicle by letting

$$[r(i,j)](E,I) = \begin{cases} 1, (i,j) \in E \cup I \\ 0, \text{otherwise} \end{cases}, (i,j) = (1,1), ...(|O|,|D|)$$

The load vector can then be inferred by letting

$$[x(i,j)](E,I) = \begin{cases} a(i,j) - \lfloor a(i,j) \rfloor, (i,j) \in E, a(i,j) \neq \lfloor a(i,j) \rfloor \\ 1, (i,j) \in E, a(i,j) = \lfloor a(i,j) \rfloor \\ 1 - \sum_{(i,j) \in E} x(i,j), (i,j) \in I \\ 0, \text{otherwise} \end{cases}, (i,j) = (1,1), ...(|O|,|D|)$$

Because the set $I$ contains at most one origin/destination pair, the vector $[x(i,j)](E,I)$ is well-defined. In order to ensure feasibility, it is necessary that $r(E,I) \in R$ and $\sum_{(i,j) \in E} x(i,j) = \sum_{(i,j) \in E} a(i,j) \leq 1$. Also, $I = \{(k,l)\}$ such that $a(k,l) > 1 - \sum_{(i,j) \in E} a(i,j)$.

The recursive equation for the DP in this form can now be written as

$$v(a) = \min_{(E,I) \in A(a,t)} \{c(E,I) + v(a - x(E,I))\} \tag{69}$$

where $A(a,t)$ is the feasible action space for state $(a,t)$.

## *6.4 Finite-State DP Formulation*

Restricting the action space to a finite collection implies that the DP can encounter a finite number of possible states. We now develop an alternate representation of the DP with finite state and action spaces, and show that the algorithm can be solved using this representation.

For an arbitrary subset $J \subseteq \{(1,1), ..., (|O|,|D|)\}$, we define $n_J$ as the lower bound on the number of routes needed to serve the service requests in $J$, such that:

$$n_J = \left\lceil \sum_{(i,j) \in J} a(i,j) \right\rceil \tag{70}$$

The vector $n(a) \in \mathbb{Z}_+^{2^{\{(1,1),...,(|O|,|D|)\} \setminus 0}}$ represents the $(2^{|O| \times |D|} - 1)$ lower bounds for all combinations for origin/destination pairs. We refer to $n(a)$ as the *n-vector*. Converting the inventory position, $a$, into this bound significantly reduces the state space. However, as will be shown, all of the information necessary for the solution of the problem is retained. Figure 18 shows an example of the relationship between $n$ and $a$ for a simple two request

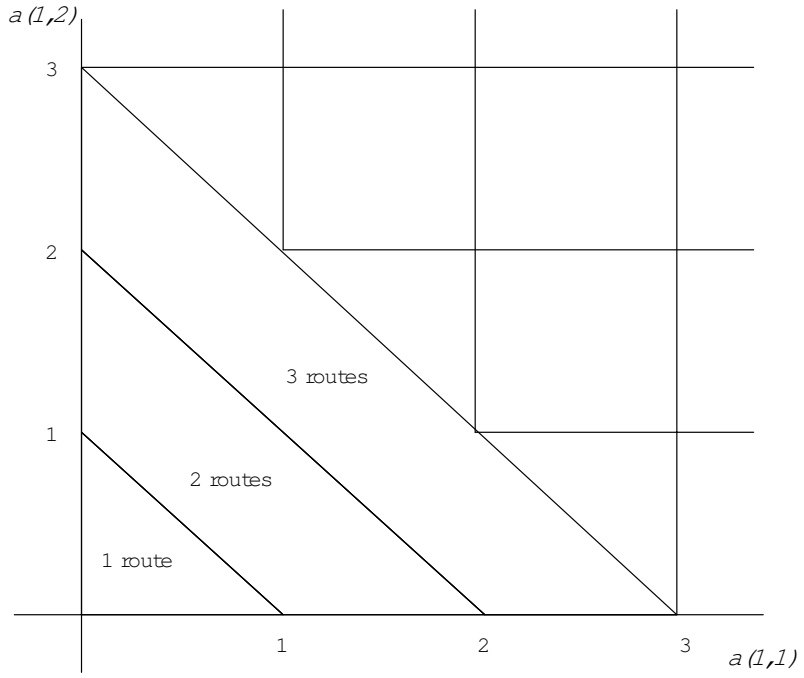


**Figure 18:** Two request graph showing invariant regions within which the cost of solution is the same for all problems.

problem, with one origin and two destinations. In this situation, when the size of the two loads requiring service sums to less than or equal to 1 truckload, only one route is required. The same is true when the two loads sum to less than or equal to 2 truckloads, and so on. The cost for servicing the two loads can be seen as invariant for each region. This invariance can be used in significantly simplifying the problem.

Given the $n$-vector $n = n(a)$ for the state $a$ and the selected action pair $(E, I)$, the $n$-vector of the resulting state, $n' = n(a - x(E, I))$, can be uniquely determined by $n(a)$ and $(E, I)$. The following theorem presents this claim. The proof, generalized from that for the SDVRP presented in Lee et al. [58], is constructive.

**Theorem 10** *Let $n = n(a)$ be the $n$-vector of the current state $a$, and let the chosen action be $(E, I)$. Then the $n$-vector of the resulting state $n' = n(a - x(E, I))$ is determined by and can be computed using only the information provided by $n$ and $(E, I)$.*

Proof: Let $U = \{1, ..., |O|\} \times \{1, ..., |D|\} \setminus (E \cup I)$ be the set of origins and destinations not serviced by the route $r(E, I)$, and let $J \subseteq \{1, ..., |O|\} \times \{1, ..., |D|\}$. The following

relationships result from the description of the sets $U$ and $E$:

- if $J \subseteq E$, then $n'_J = n_J - 1$
- if $J \subseteq U$, then $n'_J = n_J$
- if $J = \tilde{E} \cup \tilde{U}$, where $\tilde{E} \subseteq E$ and $\tilde{U} \subseteq U$, then $n'_J = n_{\tilde{U}} + n_{\tilde{E}} - 1$.

When $I = 0$, the three relationships above completely characterize all components of the vector $n'$. Suppose $I = \{(i_0, j_0)\}$. Then the following relationships allow us to describe the remaining components of $n'$:

- if $J = I = \{(i_0, j_0)\}$,

$$\begin{aligned} n'_J &= \lceil a(i_0,j_0) - x(i_0,j_0) \rceil = \left\lceil a(i_0,j_0) - (1 - \sum_{(i,j)\in E} (a(i,j) - \lfloor a(i,j) \rfloor)) \right\rceil \\ &= \left\lceil \sum_{(i,j)\in E\cup I} a(i,j) - \sum_{(i,j)\in E} \lfloor a(i,j) \rfloor - 1 \right\rceil \\ &= n_{E\cup I} - \sum_{(i,j)\in E} n'_{(i,j)} - 1 = n_{E\cup I} - (\sum_{(i,j)\in E} n_{(i,j)} - |E|) - 1 \\ &= n_{E\cup I} - \sum_{(i,j)\in E} n_{(i,j)} + |E| - 1 \end{aligned}$$

  where $|E|$ is the cardinality of the set $E$.

- if $J = \tilde{E} \cup I = \tilde{E} \cup (i_0, j_0)$, where $\tilde{E} \subseteq E$,

$$\begin{aligned} n'_J &= \left\lceil \sum_{(i,j)\in \tilde{E}\cup I} a(i,j) - x(i,j) \right\rceil = \left\lceil \sum_{(i,j)\in \tilde{E}} a(i,j) - x(i,j) + a(i_0,j_0) - x(i_0,j_0) \right\rceil \\ &= \sum_{(i,j)\in \tilde{E}} \lfloor a(i,j) \rfloor + \lceil a(i_0,j_0) - x(i_0,j_0) \rceil \\ &= \sum_{(i,j)\in \tilde{E}} \lfloor a(i,j) \rfloor - \sum_{(i,j)\in E} \lfloor a(i,j) \rfloor - 1 + \left\lceil \sum_{(i,j)\in E\cup I} a(i,j) \right\rceil \\ &= \left\lceil \sum_{(i,j)\in E\cup I} a(i,j) \right\rceil - \sum_{(i,j)\in E'} \lfloor a(i,j) \rfloor - 1 \\ &= n_{E\cup I} - \sum_{(i,j)\in E'} n'_{(i,j)} - 1 = n_{E\cup I} - \sum_{(i,j)\in E'} n_{(i,j)} + |E'| - 1 \end{aligned}$$

  where $E' \subseteq E \setminus \tilde{E}$ and $|E'|$ is the cardinality of $E'$.

- if $J = \tilde{U} \cup I = \tilde{U} \cup (i_0, j_0)$, where $\tilde{U} \subseteq U$,

$$\begin{aligned} n'_J &= \left\lceil \sum_{(i,j)\in\tilde{U}} a(i,j) + a(i_0,j_0) - x(i_0,j_0) \right\rceil \\ &= \left\lceil \sum_{(i,j)\in\tilde{U}\cup I} a(i,j) - (1 - \sum_{(i,j)\in E} (a(i,j) - \lfloor a(i,j) \rfloor)) \right\rceil \\ &= \left\lceil \sum_{(i,j)\in\tilde{U}\cup I\cup E} a(i,j) - \sum_{(i,j)\in E} \lfloor a(i,j) \rfloor - 1 \right\rceil = n_{\tilde{U}\cup I\cup E} - \sum_{(i,j)\in E} n'_{(i,j)} - 1 \\ &= n_{\tilde{U}\cup I\cup E} - \sum_{(i,j)\in E} n_{(i,j)} + |E| - 1 \end{aligned}$$

- if $J = \tilde{U} \cup \tilde{E} \cup I = \tilde{U} \cup \tilde{E} \cup (i_0, j_0)$, where $\tilde{E} \subseteq E$ and $\tilde{U} \subseteq U$,

$$\begin{aligned} n'_J &= \left\lceil \sum_{(i,j)\in\tilde{U}\cup I} a(i,j) + \sum_{(i,j)\in\tilde{E}} a(i,j) - (1 - \sum_{(i,j)\in E'} (a(i,j) - \lfloor a(i,j) \rfloor)) \right\rceil \\ &= \left\lceil \sum_{(i,j)\in\tilde{U}\cup I} a(i,j) + \sum_{(i,j)\in E} a(i,j) - 1 - \sum_{(i,j)\in E'} \lfloor a(i,j) \rfloor \right\rceil \\ &= \left\lceil \sum_{(i,j)\in\tilde{U}\cup I\cup E} a(i,j) - \sum_{(i,j)\in E'} \lfloor a(i,j) \rfloor - 1 \right\rceil \\ &= n_{\tilde{U}\cup I\cup E} - \sum_{(i,j)\in E'} n'_{(i,j)} - 1 = n_{\tilde{U}\cup I\cup E} - \sum_{(i,j)\in E'} n_{(i,j)} + |E'| - 1 \end{aligned}$$

It follows that the $n$-vector $n' = n(a - x(E, I))$ of the state resulting from choosing the action $(E, I)$ at the state $a$ is determined by and can be computed using only the information provided by $n(a)$ and $(E, I)$. □

This method was used to solve problems for the SDVRP with up to 10 customers [58]. However, in the case of the CPDPSL, the size of the $n$-vector, $(2^{|O|\times|D|} - 1)$, grows exponentially with the number of origins and destinations. The size of this vector makes it prohibitively large for use in this problem. The concept of invariance may have future potential for use in this problem, so the adaptation to the CPDPSL was included here. Reducing the action space has applicability, regardless of the formulation used. This can be seen in the following discussion concerning the algorithm used to solve the CPDPSL.

## 6.5 A* Algorithm

A forward-search shortest path algorithm was used to find an exact solution for the PDP with the constraint of origins visited before destinations on a route. The A* algorithm, a "best first" search heuristic that is guaranteed to find an optimal solution [70], was selected based on the structure of the model. As described in Pearl [73], the A* Algorithm functions as follows:

1. Put the initial state $s$ on OPEN
2. If OPEN is empty, exit with failure
3. Remove from OPEN and place on CLOSED a state $n$ for which the cost $f$ is minimum
4. If $n$ is the goal state, exit successfully with the solution obtained by tracing back the pointers from $n$ to $s$
5. Otherwise expand $n$, generating all its successor states, and attach to them pointers back to $n$. For every successor $n'$ of $n$:
   (a) If $n'$ is not already on OPEN or CLOSED, estimate $h(n')$, the cost of the best path from $n'$ to some goal state, and calculate $f(n') = g(n') + h(n')$ where $g(n') = g(n) + c(n, n')$ and $g(s) = 0$.
   (b) If $n'$ is already on OPEN or CLOSED, direct its pointers along the path yielding the lowest $g(n')$.
   (c) If $n'$ required pointer adjustment and was found on CLOSED, reopen it.
6. Go to step 2.

A* uses $h$, a lower bound on the true cost from the current state to the end state, as a guide in finding the shortest path. Each successive state is chosen such that the combination of the known cost of that state and the predicted cost to complete the solution result in a minimum cost. Therefore, the effectiveness of this algorithm is dependent on the precision of the lower bound. With perfect information, the algorithm will move directly along the

optimal path. With no information, or $h = 0$, A* reduces to Dijkstra's algorithm. The precision of the lower bound must be balanced against the complexity of calculating that bound. A good lower bound will allow the algorithm to explore a very limited number of possible solutions; however, if this bound is difficult to calculate, it may result in an increase in computational time that significantly diminishes the value of that bound.

This model uses a lower bound that is very simple to calculate; yet, the accuracy is dependent on the range of load sizes. The lower bound is the load size multiplied by the cost of travel from the depot to the origin to the destination and back to the depot. With load sizes close to vehicle capacity, this bound is tight. Otherwise, the costs may be artificially low, with little information provided to aid the algorithm.

The algorithm was coded in C. An initial state was generated by finding some combination of loads that could be picked up by the vehicle without violating capacity constraints. The rule that at most one split load can be on a route was implemented here. The cost of routing the vehicle to service these loads was determined using a simple TSP algorithm. The cost of servicing the remaining loads was determined using the lower bounding procedure. The code then followed the A* Algorithm to completion.

## *6.6 Experimental Design*

The A* algorithm was tested on problem sets of two sizes, one with 9 transportation requests and the other with 10. Each transportation request contains the origin and destination location information, and the fraction of a truckload to be delivered. Coordinates for the pickup and delivery locations were randomly generated with a uniform distribution over the range [-40,40] for both X and Y coordinates. The results below are reported with the location of the depot fixed at [0,0] for each data set. Testing with the depot moved to other locations did not alter the results. The problems with 9 transportation requests had three pickup locations and three delivery locations, while the 10 request problem had two pickup locations and five delivery locations. Three different location configurations were generated for both problem sets.

The load sizes were also randomly generated. The sizes were all less than or equal to

vehicle capacity. Without loss of generality, vehicle capacity was fixed at 1. Four ranges were used to bound the load sizes, with 10 load sets for each range. The ranges indicated the upper and lower bound (inclusive of the bound) on the fraction of the vehicle capacity that the load can occupy, and they were [0.1-0.6], [0.6-1.0], [0.3-0.6], and [0.1-1.0]. Load sizes below 0.1 were not used as splits of these loads rarely occurred and any benefit was negligible. The load sizes were randomly generated over each range with a uniform distribution. Each of the three location configurations was matched with each of the ten load sets, resulting in 30 different problems for each load range, and 120 problems overall.

The algorithm was used to solve each problem under two scenarios, with split loads and without split loads. The procedure described in the previous section was used for the algorithm with split loads. The algorithm without split loads did not allow for a split load to be added when potential load combinations were generated. All tests were run on a 2.4 GHz Xeon processor with a 400Mhz frontside bus and 2 GB RAM.

## *6.7 Experimental Results*

Table 11 presents the average percentage increase in cost without split loads, as well as the average computational time required for the algorithm with split loads. The times for the

**Table 11:** Average percentage cost increase and CPU time (in minutes) for the 9 and 10 request problems.

| Load range | 0.1 - 1.0 | | 0.1 - 0.6 | | 0.6 - 1.0 | | 0.3 - 0.6 | |
|---|---|---|---|---|---|---|---|---|
| Problem size | 9 | 10 | 9 | 10 | 9 | 10 | 9 | 10 |
| Average % cost increase | 6.68 | 5.05 | 1.57 | 1.09 | 5.66 | 8.81 | 2.99 | 2.52 |
| Average CPU minutes | 1.34 | 4.71 | 0.3 | 0.53 | 3.38 | 7.55 | 0.68 | 1.22 |

algorithm with split loads are not provided as they were all below 0.01 minutes. Figures 19 and 20 present cumulative histograms of the number of problems versus the percentage increase in cost when split loads are not allowed for the 9 and 10 request problem sets, respectively. These histograms are constructed in such a way that the farther a point is to the right on the graph, the more problems with a higher cost increase. Therefore, the lines

farthest to the right indicate those load ranges with the most cost benefit from split loads.

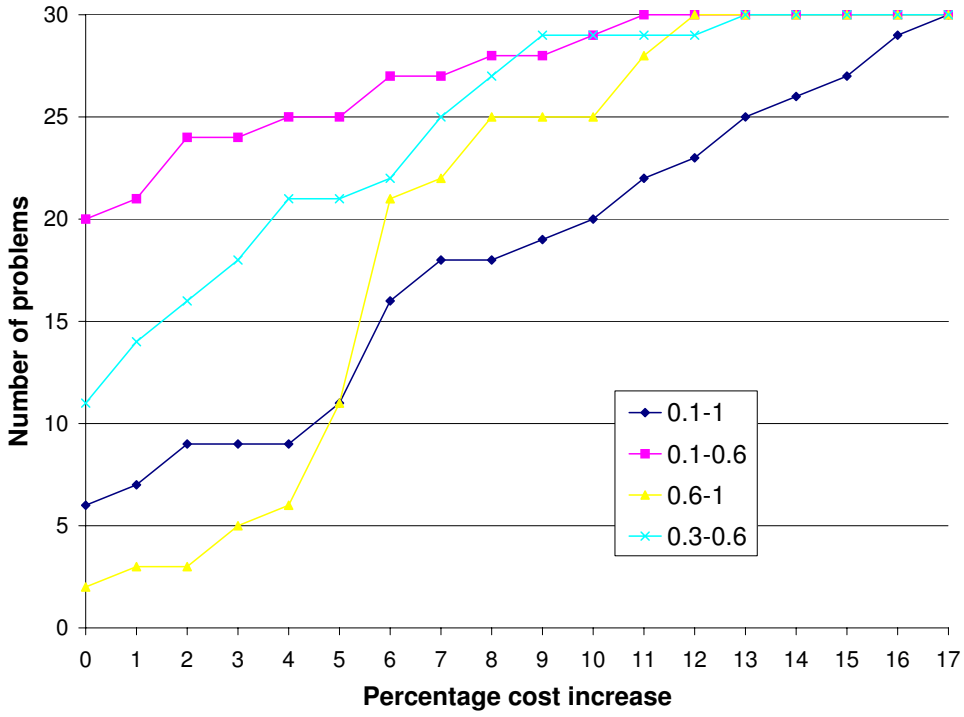


**Figure 19:** Cumulative histogram for the number of problems versus the percentage cost increase without split loads for the 9 request problem set.

The results for the exact solution of the CPDPSL are very similar to those from the heuristic solution of the PDPSL. The load ranges 0.1-1 and 0.6-1 resulted in comparable average percentage increases in cost without split loads for both solution methods. However, the benefit for the range 0.6-1 in the 10 request problem was slightly greater, while there was a more dramatic drop in benefit for the ranges 0.3-0.6 and 0.1-0.6. These discrepancies are primarily a result of the problem size and structure. For the ranges with smaller load sizes, the number of requests is such that all can be serviced with a very small number of routes, with or without split loads. Therefore, the number of routes can not be significantly reduced through the use of split loads. The cost benefit of eliminating paths to and from the depot is minimal. Also, there are fewer possibilities to consider for splitting loads. The potential for generating splits is much lower than with 75 requests.

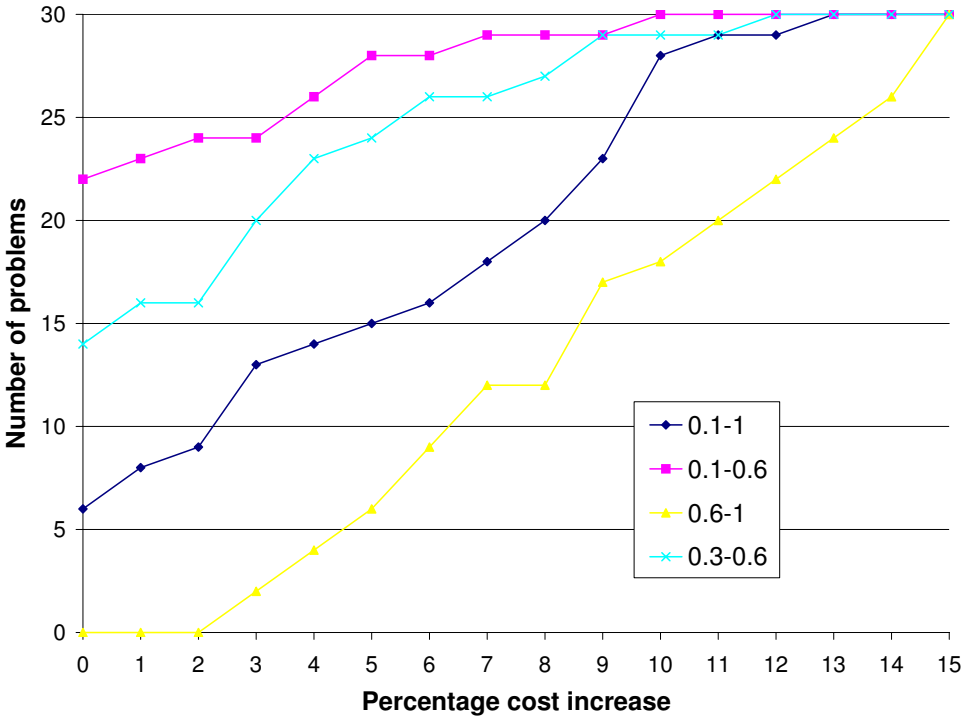


**Figure 20:** Cumulative histogram for the number of problems versus the percentage cost increase without split loads for the 10 request problem set.

The result for the 0.6-1 load size range is the converse of this as the cost benefit is greater. In this situation, all of the loads are more likely to be split and the A* algorithm is able to search all possible combinations for the optimal solution. This result suggests the potential benefit of split loads when the entire solution space is explored. An interesting result found in the 9 request problem is that the range 0.6-1 finds less benefit than the range 0.1-1. This anomaly is another result of the small problem size. The limited number of loads to deliver do not allow for a full exploration of the benefits of split loads.

The average computational times reported are not excessively long. However, there was some variation in the times within the problem sets, particularly for the load ranges with the most benefit from split loads. One 10 request problem in the 0.6-1 load range required over 40 minutes, with similar computational times for several others. These computational times prohibited testing of larger sized problem sets. While some problems with up to 15 requests could be solved in a reasonable amount of time, a thorough study of a wide range

of problems was not possible. This is also indicated by the significant increase in time from the 9 request problem size to the 10 request size for the 0.1-1 and 0.6-1 load ranges. The addition of one request more than doubles the average time in these instances.

## *6.8 Conclusions*

In this chapter, a constrained version of the Pickup and Delivery Problem with Split Loads (CPDPSL), in which all origins must be visited before any destinations on a route, was presented. The problem was formulated as a dynamic program, and several structural results were generalized from those found for the Split Delivery Vehicle Routing Problem. It was shown that the action space can be reduced by limiting the number of split loads per route to at most one. An exact solution method was presented that utilizes the A* forward-search shortest path algorithm. This algorithm was used to solve problems with up to 10 service requests. The benefit of split loads from the tests was shown to be similar to the benefit found using the PDPSL Heuristic, with some differences based on problem size and structure.

# CHAPTER VII

# CONCLUSIONS AND EXTENSIONS

## *7.1 Conclusions*

In this dissertation, we introduced the use of split loads for the Pickup and Delivery Problem. We modeled the Pickup and Delivery Problem with Split Loads (PDPSL) as a dynamic program and showed that the cost of servicing requests is monotonically increasing with the size of the loads that comprise those requests. We found an upper bound on the benefit of split loads, showing that the cost will at most double when split loads are prohibited. An alternative exact formulation was also presented, with the PDPSL as a mixed integer program.

A heuristic was developed that could solve large problems in a reasonable amount of computational time. This heuristic used elements of tabu search to generate split loads and search the solution space of the problem. A modification of the Clarke-Wright Savings algorithm was implemented to make improvements to the solution. Swaps and insertions were also utilized for further local improvements.

The heuristic was tested with randomly generated data to determine those load sizes that find the most cost benefit from split loads. These results support the structural findings, as the load sizes just above one half vehicle capacity showed the most benefit. This testing also indicated the potential for cost benefit when loads with a broad range of sizes require service, with an average benefit of 6-7% for most load sizes. It was shown that in many instances, it is computationally beneficial to limit the number of split loads per route to one. The heuristic was benchmarked with the mixed integer programming formulation. The MIP could not solve problems of a meaningful size, indicating the complexity of the problem. The heuristic did perform relatively well against the MIP with very small problem sizes, but the results indicate that there are some instances where improvements can still be made.

A case study was presented in which the PDPSL heuristic was tested on real world data. The heuristic showed the significant benefit of fleet sharing, in which multiple vehicle fleets are used to service one general pool of loads. It was also found that there is little benefit to the use of split loads under certain real world operating conditions, with a per stop cost strictly limiting any potential gain from load splitting. Furthermore, the sparse number of origin-destination pairs with loads requiring service in the data set provided little opportunity for the creation of split loads.

A constrained version of the PDPSL was presented, in which all origins must be visited prior to any destinations on a route. The problem was shown to be structurally similar to the Split Delivery Vehicle Routing Problem as several theoretical results were extended from this simpler problem. An exact algorithm was presented that was able to solve problems with up to 10 transportation requests in a reasonable amount of time. The results from this algorithm supported those found using the heuristic for the general PDPSL.

While the potential gains from the use of split loads are high, the conditions must be right in order to garner any significant benefits. Certain load sizes do not provide a lot of opportunity for load splitting, while the application of some realistic problem guidelines can considerably curtail cost benefit. This research indicates that the PDPSL warrants further exploration, given that load splitting can lead to improvements in vehicle routing and capacity utilization. However, it will be important to also study methods for overcoming the various operational hurdles to achieving these improvements.

Although this dissertation examined a range of important concepts and applications for the PDPSL, this field of study is rather new, and our research has identified a number of areas deserving of future research. We present this future work in the following section.

## *7.2 Future Work*

### 7.2.1 Model Extensions

The application of time windows to the VRP and PDP has been extensively studied. This application has not been considered in this paper; however, it is a natural extension to the PDPSL. A primary constraint in many real world problems is the restriction on when the

vehicle can make a pickup or delivery. Time windows may range from a full day to 15 minutes. In the FedEx data that was analyzed in Chapter 5, time windows were enforced. However, these time windows were generated during the routing procedure, such that they are extremely flexible to the routes that the vehicles are assigned. Many companies have more strict time windows. Adding time windows increases the level of complexity of the problem. At the same time, the additional constraints can restrict the number of solutions, allowing for a faster algorithm.

### 7.2.2 Heuristic Extensions

Applying the tabu search heuristic while using the local search moves should have significant benefit. As indicated earlier, tabu search has been shown to significantly outperform most any other metaheuristic for the Vehicle Routing Problem[35]. It has also been successfully used for the General Pickup and Delivery Problem. While some elements of this technique are used in the selection of split loads, it is not applied to the overall neighborhood search. As shown with the comparison to the mixed integer programming formulation, the heuristic can become trapped at a local minimum. A metaheuristic method such as tabu search can aid in preventing this occurrence.

The comparison with MIP also indicated that additional local search moves may be helpful. This may include the use of multiple swaps or insertions performed simultaneously. Additional moves will allow for the expansion of the search neighborhood. While this may result in an increase in computational time, it should improve the solution. Done in coordination with tabu search, both computational time and solution quality may improve.

### 7.2.3 Development of Invariant Characteristic

The invariant characteristic described for the Constrained Pickup and Delivery Problem in Chapter 6 may have some further structural or computational benefit. This characteristic, defining the number of vehicles required to service any combination of loads, is of a size that currently makes it computationally intractable for useful application to the problem. However, it has been successfully used for the Split Delivery Vehicle Routing Problem, and similar success may be found for the CPDPSL. While it does not generate a finite state

space to the same degree as with the SDVRP, it does provide a finite characteristic for an otherwise continuous problem. Analyzing the structure of this problem further may lead to an improvement of this characteristic, to the point where it may be successfully used for solving large scale problems.

### 7.2.4 Motivating Change

From an operational perspective, as indicated in the previous section, it will be important to determine how to overcome the constraints that business conditions impose on the use of split loads. Many business rules are rather rigid and difficult to change. Quantifying the benefit of split loads under current conditions and conditions with certain business rules eliminated may entice companies to make a change. Seeing the potential cost savings may have some effect on how a firm manages its transportation needs.

# REFERENCES


[1] Al-Khayyal, F. A., "Jointly constrained bilinear programs and related problems: an overview," *Computers and Mathematics with Applications*, vol. 19, pp. 53–62, 1990.

[2] American Trucking Association, "Standard trucking and transportation statistics," vol. 10, 2004.

[3] Angelelli, E. and Mansini, R., *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, ch. The Vehicle Routing Problem with Time Windows and Simultaneous Pick-up and Delivery, pp. 249–267. Springer-Verlag, 2002.

[4] Anily, S. and Mosheiov, G., "The traveling salesman problem with delivery and backhauls," *Operations Research Letters*, vol. 16, pp. 11–18, 1994.

[5] Archetti, C., Hertz, A., and Speranza, M., "A tabu search algorithm for the split delivery vehicle routing problem." Transportation Science, to appear.

[6] Archetti, C., Savelsbergh, M. W. P., and Speranza, M., "Worst-case analysis for split delivery vehicle routing problems." Transportation Science, to appear.

[7] Bander, J. L., *Sequential Decision Problems Arising in Commercial Vehicle Operations*. PhD thesis, University of Michigan, 1999.

[8] Bartholdi, J. and Platzman, L., "An O ($n \log n$) planar traveling salesman heuristic based on spacefilling curves," *Operations Research Letters*, vol. 1, pp. 121–125, 1982.

[9] Belenguer, J., Martinez, M., and Mota, E., "A lower bound for the split delivery vehicle routing problem," *Operations Research*, vol. 48, no. 5, pp. 801–810, 2000.

[10] Casco, D. O., Golden, B. L., and Wasil, E. A., *Vehicle Routing: Methods and Studies*, ch. Vehicle routing with Backhauls: models, algorithms and case studies, pp. 127–147. North-Holland, Amsterdam, 1988.

[11] Cheung, R. and Hang, D. D., "Multi-attribute label matching algorithms for vehicle routing problems with time windows and backhauls," *IIE Transactions*, vol. 35, pp. 191–205, 2003.

[12] Clarke, C. and Wright, J. Q., "Scheduling of vehicle from a central depot to a number of delivery points," *Operations Research*, vol. 12, pp. 568–581, 1964.

[13] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F., "A guide to vehicle routing heuristics," *Journal of the Operational Research Society*, vol. 53, pp. 512–522, 2002.

[14] Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., "Solution of a large-scale traveling salesman problem," *Operations Research*, vol. 2, pp. 393–410, 1954.

[15] Deif, I. and Bodin, L., "Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling," in *Proceedings of the Babson Conference on Software Uses in Transportation and Logistic Management* (Kidder, A. E., ed.), 1984.

[16] Dell'Amico, M., Righini, G., and Salani, M., "A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection." To appear, 2005.

[17] Desrosiers, J., Dumas, Y., and Soumis, F., "A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows," *American Journal of Mathematical and Management Sciences*, vol. 6, pp. 301–325, 1986.

[18] Dethloff, J., "Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up," *OR Spektrum*, pp. 79–96, 2001.

[19] Dethloff, J., "Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls," *Journal of the Operational Research Society*, pp. 115–118, 2002.

[20] Dror, M., Laporte, G., and Trudeau, P., "Vehicle routing with split deliveries," *Discrete Applied Mathematics*, vol. 50, pp. 239–254, 1994.

[21] Dror, M. and Trudeau, P., "Savings by split delivery routing," *Transportation Science*, vol. 23, pp. 141–145, 1989.

[22] Dror, M. and Trudeau, P., "Split delivery routing," *Naval Research Logistics*, vol. 37, pp. 383–402, 1990.

[23] Duhamel, C., Potvin, J. Y., and Rousseau, J. M., "A tabu search heuristic for the vehicle routing problem with backhauls and time windows," *Transportation Science*, vol. 31, pp. 49–59, 1997.

[24] Dumas, Y., Desrosiers, J., and Soumis, F., "The pickup and delivery problem with time windows," *European Jounal of Operational Research*, vol. 54, pp. 7–22, 1991.

[25] Federgruen, A. and Simchi-Levi, D., *Network Routing*, vol. 8 of *Handbooks in OR and MS*, ch. Analysis of vehicle routing and inventory routing problems, pp. 297–373. Amsterdam, The Netherlands: North-Holland, 1995.

[26] Feitler, J. N., Corsi, T. M., and Grimm, C. M., "Measuring firm strategic change in the regulated and deregulated motor carrier industry: An 18 year evaluation," *Transportation Research: Part E, Logistics and Transportation Review*, vol. 33, pp. 159–169, 1997.

[27] Fischetti, M. and Toth, P., "An additive bounding procedure for combinatorial optimization problems," *Operations Research*, vol. 37, pp. 319–328, 1989.

[28] Fisher, M. and Jaikumar, R., "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, pp. 109–124, 1981.

[29] Frizzell, P. W. and Giffin, J. W., "The bounded split delivery vehicle routing problem with grid networks distances," *Asia Pacific Journal of Operational Research*, vol. 9, pp. 101–116, 1992.

[30] Frizzell, P. W. and Giffin, J. W., "The split delivery vehicle scheduling problem with time windows and grid network distance," *Computers and Operations Research*, vol. 22, pp. 655–667, 1995.

[31] Gèlinas, S., Desrochers, M., Desrosiers, J., and Solomon, M. M., "A new branching strategy for the time constrained routing problem with application to backhauling," *Annals of Operations Research*, vol. 61, pp. 91–110, 1995.

[32] Gendreau, M., Hertz, A., and Laporte, G., "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol. 40, pp. 1276–1290, 1994.

[33] Gendreau, M., Hertz, A., and Laporte, G., "The traveling salseman problem with backhauls," *Computers and Operations Research*, vol. 23, pp. 501–508, 1996.

[34] Gendreau, M., Laporte, G., and Hertz, A., "An approximation algorithm for the traveling salseman problem with backhauls," *Operations Research*, vol. 45, pp. 639–641, 1997.

[35] Gendreau, M., Laporte, G., and Potvin, J.-Y., *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*, ch. Metaheuristics for the capacitated VRP, pp. 129–154. SIAM Publishing, Philadelphia, 2002.

[36] Gendreau, M., Laporte, G., and Vigo, D., "Heuristics for the traveling salesman problem with pickup and delivery," *Computers and Operations Research*, vol. 26, pp. 699–714, 1999.

[37] Glover, F., "Future paths for integer programming and links to artificial intelligence," *Computers and Operations Research*, vol. 5, pp. 533–549, 1986.

[38] Glover, F., "Tabu search - part I," *ORSA Journal on Computing*, vol. 1, pp. 190–206, 1989.

[39] Glover, F., "Tabu search - part II," *ORSA Journal on Computing*, vol. 2, pp. 4–32, 1990.

[40] Goetschalckx, M. and Jacobs-Blecha, C., "The vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. 42, pp. 39–51, 1989.

[41] Goetschalckx, M. and Jacobs-Blecha, C., "The vehicle routing problem with backhauls: Properties and solution algorithms," Tech. Rep. MHRC-TR-88-13, Georgia Institute of Technology, 1993.

[42] Golden, B. L. and Assad, A. A., eds., *Vehicle Routing: Methods and Studies*. North-Holland, Amsterdam, 1988.

[43] Gutin, G. and Punnen, A. P., eds., *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[44] Halse, K., *Modeling and solving complex vehicle routing problems*. PhD thesis, IMSOR, The Technical University of Denmark, 1992.

[45] Healy, P. and Moll, R., "A new extension of local search applied to the dial-a-ride problem," *European Journal of Operational Research*, vol. 83, pp. 83–104, 1995.

[46] Ho, S. C. and Haugland, D., "A tabu search heuristic for the vehicle routing problem with time windows and split deliveries," *Computers and Operations Research*, vol. 31, pp. 1947–1964, 2004.

[47] Hoffman, M., Hough, L., and Nowak, M., "The package express industry: a historical and current perspective." To appear as a chapter in *Trucking in the Age of Information*, Ashgate: Aldershot, Hampshire, UK, 2005.

[48] Holland, J. H., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[49] Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M. M., and Villeneuve, D., "A request clustering algorithm for door-to-door handicapped transportation," *Transportation Science*, vol. 29, no. 1, pp. 63–78, 1995.

[50] Jaw, J.-J., Odoni, A. R., Psaraftis, H. N., and Wilson, N. H. M., "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows," *Transportation Research Part B*, vol. 20B, pp. 243–257, 1986.

[51] Kalantari, B., Hill, A. V., and Arora, S. R., "An algorithm for the traveling salesman problem with pickup and delivery customers," *European Journal of Operational Research*, vol. 22, pp. 377–386, 1985.

[52] Kirkpatrick, S., Gellatt Jr., C. D., and Vecchi, M. P., "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[53] Landrieu, A., Mati, Y., and Binder, Z., "A tabu search heuristic for the single vehicle pickup and delivery problem with time windows," *Journal of Intelligent Manufacturing*, vol. 12, pp. 497–508, 2001.

[54] Laporte, G. and Osman, I. H., "Routing problems: A bibliography," *Annals of Operations Research*, vol. 61, pp. 227–262, 1995.

[55] Laporte, G. and Semet, F., *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*, ch. Classical heuristics for the capacitated VRP, pp. 109–128. SIAM Publishing, Philadelphia, 2002.

[56] Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., and Shmoys, D. B., eds., *The Traveling Salesman Problem: A Guidede Tour of Combinatorial Optimization*. John Wiley and Sons, Inc., 1985.

[57] Lee, B., "Building the real information superhighway," *Red Herring*, p. 66, February 2002.

[58] Lee, C., White, C., Bozer, Y., and Epelman, M., "A shortest path approach to the multiple-vehicle routing problem with split pick-ups." to appear in Transportation Research - Part B, 2005.

[59] Little, J., Murty, K., Sweeney, D., and Karel, C., "An algorithm for the traveling salesman problem," *Operations Research*, vol. 11, pp. 972–989, 1963.

[60] Lu, Q. and Dessouky, M., "An exact algorithm for the multiple vehicle pickup and delivery problem," *Transportation Science*, vol. 38, pp. 503–514, 2004.

[61] Min, H., "The multiple vehicle routing problem with simultaneous delivery and pickup points," *Transportation Research-A*, vol. 23A, pp. 377–386, 1989.

[62] Mingozzi, A., Giorgi, S., and Baldacci, R., "An exact method for the vehicle routing problem with backhauls," *Transportation Science*, vol. 33, pp. 315–329, 1999.

[63] Mitrović-Minić, S., "Pickup and delivery problem with time windows: A survey," tech. rep., Simon Fraser University, 1998.

[64] Morlok, E. K., Nitzberg, B. F., and Balasubramaniam, K., "The parcel service industry in the U.S.: Its size and role in commerce," tech. rep., University of Pennsylvania, 2000.

[65] Mosheiov, G., "The traveling salesman problem with pick-up and delivery," *European Journal of Operational Research*, vol. 79, pp. 299–310, 1994.

[66] Mullaseril, P. A., Dror, M., and Leung, J., "Split-delivery routeing heurisitcs in livestock feed distribution," *Journal of the Operational Research Society*, vol. 48, pp. 107–116, 1997.

[67] Nagy, G. and Salhi, S., "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries," *European Journal of Operational Research*, vol. 162, pp. 126–141, 2005.

[68] Nanry, W. P. and Barnes, J. W., "Solving the pickup and delivery problem with time windows using reactive tabu search," *Transportation Research Part B*, vol. 34, pp. 107–121, 2000.

[69] Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization*. John Wiley and Sons, Inc., 1999.

[70] Nilsson, N., *Principles of Artificial Intelligence*. Tioga, 1980.

[71] Nowak, M. A., Chelsea C. White, I., and Hiller, B., "The impact of removing jurisdictional constraints on paratransit fleet efficiency," tech. rep., Ann Arbor Transportation Authority, 1999.

[72] Osman, I. H. and Wassan, N. A., "A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls," *Journal of Scheduling*, vol. 5, pp. 263–285, 2002.

[73] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Co., 1984.

[74] Potvin, J.-Y., Duhamel, C., and Guertin, F., "A genetic algorithm for vehicle routing problem with backhauling," *Applied Intelligence*, vol. 6, pp. 345–355, 1996.

[75] Prins, C., "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers and Operations Research*, vol. 31, pp. 1985–2002, 2004.

[76] Psaraftis, H. N., "A dynamic programming solution to the single vehicle many-to-many immeditate request dial-a-ride problem," *Transportation Science*, vol. 14, no. 2, pp. 130–154, 1980.

[77] Psaraftis, H. N., "Analysis of an $O(N^2)$ heuristic for the single vehicle many-to-many euclidean dial-a-ride problem," *Transportation Research Part B - Methodological*, vol. 17B, pp. 133–145, 1983.

[78] Psaraftis, H. N., "An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows," *Transportation Science*, vol. 17, no. 3, pp. 351–357, 1983.

[79] Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics, New York: John Wiley and Sons, Inc., 1994.

[80] Renaud, J., Boctor, F. F., and Laporte, G., "Perturbation heuristics for the pickup and delivery traveling salesman problem," *Computers and Operations Research*, vol. 29, pp. 1129–1141, 2002.

[81] Renaud, J., Boctor, F. F., and Ouenniche, J., "A heuristic for the pickup and delivery traveling salesman problem," *Computers and Operations Research*, vol. 27, pp. 905–916, 2000.

[82] Ruland, K. S. and Rodin, E. Y., "The pickup and delivery problem: Faces and branch-and-cut algorithm," *Computers and Mathematics with Applications*, vol. 33, pp. 1–13, 1997.

[83] Salhi, S. and Nagy, G., "A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling," *Journal of the Operational Research Society*, vol. 50, pp. 1034–1042, 1999.

[84] Savelsbergh, M. and Sol, M., "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995.

[85] Savelsbergh, M. and Sol, M., "Drive: Dynamic routing of independent vehicles," *Operations Research*, vol. 46, pp. 474–490, 1998.

[86] Sexton, T. R. and Bodin, L. D., "Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling," *Transportation Sciences*, vol. 19, pp. 378–410, 1985.

[87] Sexton, T. R. and Bodin, L. D., "Optimizing single vehicle many-to-many operations with desired delivery times: II. routing," *Transportation Sciences*, vol. 19, pp. 411–435, 1985.

[88] Sexton, T. R. and Choi, Y.-M., "Pickup and delivery of partial loads with 'soft' time windows," *American Journal of Mathematical and Management Sciences*, vol. 6, pp. 369–398, 1986.

[89] Sierksma, G. and Tijssen, G. A., "Routing helicopters for crew exchanges on offshore locations," *Annals of Operations Research*, vol. 76, pp. 261–286, 1998.

[90] Taillard, E., "Parallel iterative search methods for vehicle routing problems," *Networks*, vol. 23, pp. 661–673, 1993.

[91] Toth, P. and Vigo, D., "An exact algorithm for the vehicle routing problem with backhauls," *Transportation Science*, vol. 31, no. 4, pp. 372–385, 1997.

[92] Toth, P. and Vigo, D., "Heuristic algorithms for the handicapped persons transportation problem," *Transportation Science*, vol. 31, no. 1, pp. 60–71, 1997.

[93] Toth, P. and Vigo, D., "A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls," *European Journal of Operational Research*, vol. 113, pp. 528–543, 1999.

[94] Toth, P. and Vigo, D., "Models, relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics*, vol. 123, pp. 487–512, 2002.

[95] Toth, P. and Vigo, D., "The granular tabu search and its application to the vehicle routing problem," *INFORMS Journal on Computing*, vol. 15, pp. 333–346, 2003.

[96] Van der Bruggen, L. J. J., Lenstra, J. K., and Schuur, P. C., "Variable-depth search for the single-vehicle pickup and delivery problem with time windows," *Transportation Science*, vol. 27, pp. 298–311, 1993.

[97] Wade, A. C. and Salhi, S., "An investigation into a new class of vehicle routing problem with backhauls," *Omega*, vol. 30, pp. 479–487, 2002.

[98] Xu, J. and Kelly, J., "A network flow-based tabu search heuristic for the vehicle routing problem," *Transportation Science*, vol. 30, pp. 379–393, 1996.

[99] Yano, C., Chan, T., Richter, L., Cutler, T., Murty, K., and McGettigan, D., "Vehicle routing at quality stores," *Interfaces*, vol. 17, pp. 52–63, 1987.