2009

A Metaheuristic for the Pickup and Delivery Problem with Split-Loads and its Extension

Dai YAO
Singapore Management University, dai.yao.2007@mom.smu.edu.sg

A Metaheuristic for the Pickup and Delivery Problem with Split-Loads and its Extension

by

Dai YAO

Singapore Management University

2009

A Metaheuristic for the Pickup and Delivery Problem with Split-Loads and its Extension

by

Dai YAO

Submitted in partial fullfillment

of the requirements for the Degree of

Master of Science in Operations Management

Singapore Management University

2009

This thesis entitled:

A Metaheuristic for the Pickup and Delivery Problem with Split-Loads and its Extension

written by Dai YAO

has been approved by the Lee Kong Chian School of Business

Assoc. Prof. Brian Rodrigues

Practise Assoc. Prof. Moosa Sharafali

Assis. Prof. Lim Yun Fong

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

A Metaheuristic for the Pickup and Delivery Problem with Split-Loads and its Extension-Abstract

by

Dai YAO

In this dissertation, we study improvements in the Pickup and Delivery Problem that can be achieved by allowing multiple vehicle trips to serve a common load. We explore how costs can be reduced through the elimination of the constraint that a load must be served by only one vehicle trip. Specifically, we investigate the problem of routing vehicles to serve loads that have distinct origins and destinations, with no constraint on the amount of a load that a vehicle may serve at a time.

We develop a metaheuristic to solve large scale practical size problems in this form and apply the metaheuristic to randomly generated data sets. The metaheuristic is based on a predetermined fixed number of restarts of annealing-like procedure with tabu-lists to avoid cycling in the search process and the annealing-like procedure is to guide the local search in three neighborhoods defined to solve the problem. We test the algorithm on several sets of problem instances generated with different transportation requests and over different load size ranges. The experimental results on these problem sets have shown that benefits are common if split loads are adopted in designing practical sized transportation network for different load size configurations, and the most benefit is achieved when all the loads are just a little above half of the vehicle capacity and have small variations, and this most benefit is around 33% for all the three 75-, 100-, and 125-request problem sets, which overtakes the one reported in previous literature. In a more general setting when some load sizes are greater than the vehicle capacity and

have to be split, there are also certain cost reduction if split loads are applied. We also generate numeral tests on different load size ranges and split the loads that are greater than the vehicle capacity using different "splitting" strategy, in term of how much amount to split from the original load to form a new load, and find that there seem to be no optimal "splitting" strategy, which can assure the best quality of solutions using the metaheuristic developed in the dissertation.

Contents

Chapter

Acknowledgements

It was a great honor for me to be picked up as one of the first batch of Master of Science in Operations Management students in Singapore Management University. While working on my MSc degree here, I have received a great amount of support and help from different people. My degree would be impossible without these support and help. I would like to begin by thanking my independent study supervisor and dissertation advisor, Dr. Brian Rodrigues. With other faculty members, He opened the door for me to the Operations Management area, and he constantly and well guided me through the whole process and always encouraged me to challenge myself to even higher height.

I also would like to thank the OM faculty members at Singapore Management University and people from elsewhere, who have been helping a lot in my transformation from an Engineering mindset to an Operations Management mindset. In particular, I would like to thank Dr. Lim Yun Fong, Dr. Moosa Sharafari, Dr. Wee Kwan Eng, Dr. Wu Zhengping, Dr. Onur, at SMU, and Dr. Mahmut Palar from McMaster University in Canada, for conducting me whether compulsory basic courses or seminar courses. I would also like to thank Dr. Ding Qing for guiding me how to find interesting research problems and how to do research, and I am really grateful to Dr. Adam Zhu, Dr. Li Rong, Dr. Wang Siqun, Dr. Pascale Crama and the people mentioned above for their kind and useful suggestions regarding my current status and prospect ahead. I would like also to thank

Dr. Moosa Sharafali and Dr. Lim Yun Fong to serve in my dissertation committee and provide many needed suggestions. Meanwhile, I would like to thank Dr. Jeremy Goh from Finance department at SMU and Dr. Reddi Kotha from Management department at SMU for providing great research opportunities so I could also touch upon other research stuff in other disciplines in business management besides Operations Management, and helping me form the diversified research mindset.

I also owe a great deal 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 with any problem. And I am especially indebted to these friends who helped me with my applications to the world famous business schools for a further Ph.D study in my chosen discipline. There are so many people to mention here, and I hope I am not leaving any one alone: Lee Yen Teik, Lin Leming and Chen Fang from Lee Kong Chian School of Business at SMU, Benjamin EE & Luo Wei from Fuqua School of Business at Duke University, Zhang Kaifu from INSEAD Business School, Zhang Kaicheng from Kenan-Flagler School of Business at UNC at Chapel Hill, Wu Xiaole from Olin School of Business at Washington University at St. Louis, Li Jun from Wharton Business School at University of Pennsylvania. And I also thank my previous school mates in Tsinghua Unviersity, such as Chen Xingrong, He Zihao, Liu Xiaokang, Huang Yaming, Luo Ji and old friends such as Guo Ruifeng, Zhang Yan, Lin Xiaotian, Qian Yanjiang, Zhang Xiaoqin for their continues support and concern which helped me overcome the difficulty of losing my laptop before certain key date and other difficulties during my working towards my master degree.

Finally, I would like to thank my family. Without their continuous support and care I would never have finished the journey. Thanks to Dad and Mom for their regular cares and encouragement, and to my older sister and her husband

for always encouraging me that I am clever enough to pursue any degree I am interested.

Dedication

To the people whom I love and who love me.

Tables

Table

Figures

Figure

Chapter 1

Introduction

Transportation companies are always struggling to eliminate costs due to endogenous and exogenous dynamics to gain a cutting edge over its market rivals. From a transportation company's perspective, the endogenous dynamics it constantly faces are mainly related to its resource defaults like vehicles broken down, unavailability of a driver simply because he is ill, etc., which are unpredictable and hard to control. Transportation companies usually employ excess resources (more vehicles and more drivers) to cope with such unhappy situations. The exogenous dynamics, however, if handled properly, can be effectively avoided or hedged to a certain extend, e.g. the fluctuations of oil prices can be hedged by strategically buying oil options in financial markets, the gradually exposed demand information between different locations the company is operating can be well handled by adopting effective vehicle routing algorithms in a rolling-horizon fashion, which is called online/real-time vehicle routing. These controllable dynamics are of interest to both academic researchers and industry practitioners.

A transportation company usually has a fleet of vehicles to serve demands and has operations in a complex network consisting of many sites, and these demands may have or not associated constraints, e.g. service quantity regarding the loads to be transported, completion time requirements. The transportation company may have some constraints itself as well, such as time window constraints

at each of the sites, the maximum driving time for each of the drivers who operate the fleet of vehicles, the maximum length of a route that can be managed, etc. A vehicle routing algorithm usually takes some of the constraints into consideration and try to effectively route a vehicle and select the loads to be placed on the vehicle, whether through consolidating different loads together or split a load into different pieces, to take advantage of the vehicle time and capacity, which has been studied a lot in the Vehicle Routing Problem(VRP) literature and there are many variants corresponding to different combinations of problem types and constraints.

The dissertation focuses on improvements in vehicle routing that can be gained by allowing multiple vehicles, or multiple visits by an identical vehicle, to serve a common load, and each load is general, i.e. has no specific less than, greater than, or equal to numerical relationship with vehicle capacity, which is normalized to be 1, and we start off by studying a simplified version of the problem where all the loads are capped at the vehicle capacity, i.e. less than or equal to the vehicle capacity. We explore the benefit of using split loads in this setting and then explore the role of split loads in more general setting. We also quantify the benefit of employing split loads in the simplified problem setting and describe those instances that lead to the most benefit. Subsequently, we extend the simplified problem to the case when all the loads are general, and explore different splitting strategies, in terms of the amount of load to split to form a new load, and find that there seems to be no optimal splitting strategy that can always guarantee the best quality of solutions using the metaheuristic developed in the dissertation.

1.1 Overview

Transportation companies nowadays are forced to operate very meticulously because of the oil surge, tighter regulations on driving hours and conditions, and

higher payment and increasing shortages of driver sources. While the external pressures are growing, competition within the transportation industry is also turning fierce, with the number of companies in the business is continuously increasing [1]. However, despite various cost saving measures having been implemented at the strategic level, such as entering in certain oil options contract to hedge the fluctuation of oil price, renting vehicles/drivers to cope with determined routes instead of using dedicated ones, building transportation alliance, etc., transportation companies continue to suffer from inefficiency. One of the major indicators of the 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. Each of these occurrences represents a potential room to shrink costs.

An obvious solution to the excess capacity problem is to consolidate partial (or full) loads to be transported on similar routes onto common vehicles. If a supplier is providing parts for two manufacturers in the adjacency, using one vehicle for both deliveries is clearly more efficient than dedicating separate vehicles. Similarly, a vehicle returning on an empty backhaul may be used to transport a load intended to travel close to its return site. 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 afford together 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 benefits are not.

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 benet 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 ([2], [3], [4], [5]). Recent study on the Pickup and Delivery Problem (PDP), in which each load has a specific origin and destination associated with it, has shown that there is also huge benefit of using split loads for the problem [6]. And it has been proven that the benefit of load splits for PDP is maximized when all the loads a a little more than half of the vehicle capacity, and the authors quantify that the maximum cost saving of using split load is half of the original cost when no split load allowed. The authors also verify the conclusion by numerical examples and a case study from a real business.

Figure 1.1 from [6] presents an example of the potential benefit. The load sizes represent the portion of a truckload 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.

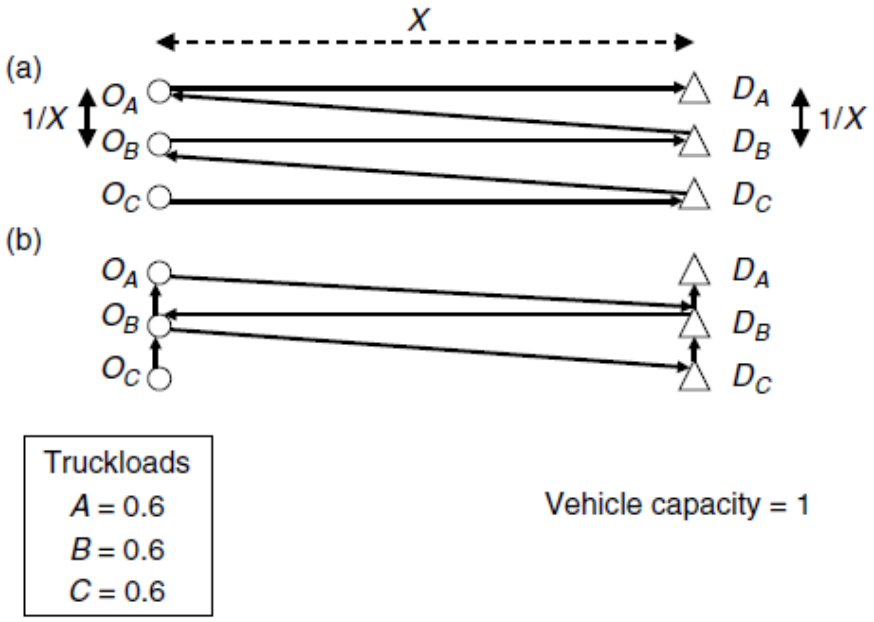


Figure 1.1: Example of benefit of split loads showing a) route plan without split loads; and b) route plan with split loads. Initially constructed as Figure 1 in [6].

The next section will introduce the outline of the dissertation and later we will describe the pickup and delivery problem in details in this chapter.

1.2 Dissertation Outline

In the remainder of this chapter, we will present a general description of the Pickup and Delivery Problem with Split Loads, as well as discuss the relevant literature of the problem studied in the dissertation.

Chapter 2 is devoted to mathematical formulation and notation introduction. Section **??** cites from [6] a mathematical formulation of the Pickup and Delivery Problem with Split Loads. Section 2.1 shows some structural properties of the problem, some of which are from existing literatures.

Chapter 3 solves the Pickup and Delivery Problem with Split Loads and all the loads are less than or equal to the vehicle capacity. Section 3.1 introduces

the way to construct an initial solution, Section 3.2 introduces four different local search methods to explore different neighborhoods of a solution, and Section 3.3 presents the general metaheuristic structure. Section 3.4 concludes and closes the chapter.

Chapter 4 presents numerical tests for the problem described in Chapter 3. Section 4.1 describes the general experimental design, including how the desired data is generated and the number of problem sets that are generated and tested. Section 4.2 presents the experimental results which consist of three main experiments, Section 4.2.1 presents the relationship between cost reduction and the maximum allowed number of splits on a load, Section 4.2.2 presents the results with loads generated over smaller load size ranges, and Section 4.2.3 presents the results with loads generated over wider load size ranges. Section 4.3 concludes and closes the chapter.

Chapter 5 uses numerical tests to study the Pickup and Delivery Problem with Split Loads when some loads are greater than the vehicle capacity. Section 5.1 describes the general experimental design. Section 5.2 presents the major experimental results, and Section 5.3 concludes and closes the chapter.

Chapter 6 concludes the dissertation and discuss future possible research questions.

1.3 Description of the Pickup and Delivery Problem with Split-Loads

The Pickup and Delivery Problem with Split Loads is originated in [6], and it 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 [10]. In general, a load 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 best policy for selecting the next site 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 there are many loads at the origin that are to be dropped at multiple destinations, the vehicle may pick up several loads at the same time. 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.

There are two conditions under which a load is considered to be split, which are first defined in Nowak et. al. 2008 [6]:

(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 pick 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 3.3 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, then it is considered to be split.

1.4 Literature Review

The PDP is the root of and most relevant to the work presented here, however, due to the intrinsic relations between PDP and VRP, that VRP is a special case of PDP that all the origins or destinations are a central depot, and as split loads have been most extensively applied to the VRP, we also consider VRP as a most relevant topic. Laporte (2007) provides a thorough review of the general vehicle routing problem, Bräysy and Gendreau (2005a, 2005b) provide an excellent review of the vehicle routing problem with time windows, and Savelsbergh and Sol (1995) provides the thorough review of the earlier work on the general pickup and

delivery problem. We have mentioned already in Section 1.1 that the only work on PDPSL existing in the literature is Nowak, Ergun and White (2008) [6], leaving only SDVRP[1] , SDVRPTW[2] and PDPTW[3] to review. We are going to review these three topics sequentially in the following two sub-sections. In Section 1.4.1, we review the pertinent literature of SDVRP and SDVRPTW while in Section 1.4.2, we review the related topics of PDPTW.

1.4.1 Review of SDVRP & SDVRPTW

In the SDVRP, the single visit assumption is relaxed and each customer may be served by more than one vehicle. Compared with other variants of the VRP, The SDVRP has received little attention in the past. It is first introduced by Dror and Trudeau (1989). They define the SDVRP problem, derive some important structural properties and propose a local search heuristic to show how split deliveries could result in savings, both in the total distance traveled and the number of vehicles utilized. As shown by Dror and Trudeau (1990), the VRPSD is a relaxation of the classical vehicle routing problem, but it remains NP-hard. Dror, Laporte, and Trudeau (1994) presents an integer programming formulation to the problem and develops an exact constraint relaxation branch and bound algorithm for the VRPSD. Frizzell and Giffin (1995) studies the problem with grid network distances and time windows constraints. Their proposed heuristics are especially tailored for the problems and this kind of network structure. Mullaseril, Dror and Leung (1997) presents a heuristic, similar to the one proposed by Dror and Trudeau (1989, 1990), for the split-delivery capacitated rural postman problem with time windows on arcs, and the heuristic is applied to a real-life problem of managing the trucks for distributing feed in a cattle ranch in Arizona. Sierksma

[1] Vehicle routing problem with split deliveries
[2] Vehicle routing problems with split deliveries and time windows
[3] Pickup and delivery problems with time windows

and Tijssen (1998) also study some real application.

In some recent work, Belenguer, Martinez and Mota (2000) propose a lower bound and computationally test this bound. Archetti, Savelsbergh, and Speranza (2006) also give a tight bound on the cost reduction that can be obtained by allowing split deliveries. Ho and Haugland (2004) present a tabu search heuristic that takes time windows into consideration and applies four common move operators while simultaneously generating split loads. Archetti, Hertz, and Speranza (2006) also employ a tabu search heuristic to solve the SDVRP. They use insertion moves in local improvement, with the possibility of inserting a customer into a route without removing the customer from another route. In Archetti, Speranza and Savelsbergh (2008), they further the idea, and present a solution approach that integrates tabu search heuristic with optimization by using an integer program to explore promising parts of the search space identified by the tabu search heuristic. This approach improves the solutions of their previously developed heuristic (Archetti, Hertz, and Speranza 2006) in all but only one instance of a large test set.

1.4.2 Review of PDPTW

Most existing works focus on the single vehicle pickup and delivery problem with time windows (1-PDPTW) due to difficulty of PDPTW. Psarafits (1980, 1983) develop a dynamic programming algorithm with $O(n^2 3^n)$ time complexity which can solve problems with up to 10 requests. Sexton and Bodin (1985a, 1985b) solve the 1-PDPTW by breaking it into a coordinating routing master problem formulated as an integer program, and a scheduling subproblem for a fixed route, which was formulated as linear program. By using a heuristic developed from Bender's decomposition, the routing master problem and the scheduling subprogram were solved individually. Sexton and Choi (1986) used a similar approach

to minimize a linear combination of total vehicle operating time and total customer penalty due to the violation of the time windows for the single vehicle pickup and delivery problem with soft time windows. In another paper, Sexton and Bodin (1983) present an insertion algorithm for the 1-PDPTW. Desrosiers et al. (1986) use a forward dynamic programming for the single vehicle dial-a-ride problem. They minimize the total travel costs considering both the time and distance dimension when solving the shortest path problem with time windows. The efficiency of the algorithm is improved by eliminating states that are incompatible with vehicle capacity, precedence and time window constraints. Van der Bruggen et al.(1993) presented a two-phase heuristic method based on arc-exchange procedures and an alternative algorithm based on simulated annealing. The approaches produce near-optimal solutions on 38 real-life problems in less than 150 seconds each. Healy and Roll (1995) propose a new local search extension called sacrificing, yielding significant improvements without deterioration of the running time, to resolve problems of size from 10 to 100 customers. Recently, Landrieu, Mati and Binder (2001) present a tabu search heuristic uses local improvements. The vertices are swapped or inserted within a route, while respecting precedence and vehicle capacity constraints, to construct the neighborhood.

The multiple vehicle pickup and delivery problem with time windows has received less attention. Dumas, Desrosiers and Soumis (1986) propose an optimal algorithm which adopts a column generation scheme with a shortest path subproblem with capacity, time window, precedence and coupling constraints. The algorithm can solve 1-PDPTW problems with up to 55 requests and multiple vehicle PDPTW with smaller number of requests per vehicle. Nanry and Barnes (2000) propose a reactive tabu search heuristic in which origin and destinations may be either inserted from one route to another or swapped with an origin-destination pair on another route to generate new solutions, with a third type of

move is to insert individual origins or destinations forward or backward within a route to improve the solution quality or feasibility, as these two moves may violate vehicle capacity constraints. Similarly, Toth and Vigo (1997) present a parallel insertion heuristic to solve the handicapped persons transportation problem. Li and Lim (2001) develop a tabu embedded simulated annealing algorithm which restarts a search procedure from the current best solution after several non-improving search iterations. They also generate six new data sets consisting of 56 100-customer problem instances from Solomon (1987)'s benchmark instances for the vehicle routing problem with time windows. The computational tests have shown this algorithm can handle practical sized multiple vehicle PDPTW problem instances with various distribution properties. Lim, Lim, and Rodrigues (2002) apply squeaky wheel optimization and local search to the PDPTW. Bent and Hentenryck (2006) develop a similar algorithm for the PDPTW, in which a simulated annealing approach is used to decrease the number of routes and a large neighborhood search is used to decrease total travel cost. Xu et al. (2003) consider a PDPTW with several extra real-life constraints, including multiple time windows, compatibility constraints, and maximum driving-time restrictions and propose a column generation heuristic to solve the practical problem. Their algorithm can solve problem instances with up to 500 requests. Ropke and Pisinger (2006) develop an adaptive large neighborhood search heuristic to solve the PDPTW, which combines a number of competing subheuristics that are used with a frequency corresponding to their historic performance and is able to improve the solutions greatly of more than 350 benchmark instances with up to 500 requests from Li and Lim (2001).

1.5 Contributions

In this dissertation, we investigate the Pickup and Delivery Problem with Split-Loads and develop a metaheuristic to solve the problem, which restarts from the previous best solution after a predetermined number of non-improving search iterations. We show by numerical tests that the most cost reduction is achieved when all the load demands are only a little bit more than half of the vehicle capacity and with small variations, and the most cost reduction is theoretically approaching half of the original cost when split loads are not allowed, while in the numerical tests, the cost reduction in most cases are far below the best result due to realistic factors such as location proximities, demand variations, among others.

We also explore the extension to the basic PDPSL problem by generalizing the load, meaning that the amount of each load can be either less than, or equal to, or greater than the capacity of the vehicle to deliver the loads. The generality of a load is commonly encountered in reality. And we find that in general there are some benefits by applying split loads to the transportation network design even when the loads are general and those loads that are greater than the vehicle capacity have to be split, and we find there is no optimal splitting policy that can always guarantee the best quality of solutions.

Chapter 2

Properties of the Problem

We have described the Pickup and Delivery Problem with Split Loads in Chapter 1.1. In this chapter, we present some structural properties of the problem that have been introduced in previous literatures; A non-linear formulation, and the only formulation of the problem, can be found in [7].

2.1 Properties of the Problem

In this section, we present three properties of the problem. The first two structural properties were found and introduced in [6], describing how much is the benefit by the use of split loads in the pickup and delivery problem and the conditions when the benefit is maximized, while the last one describes the complexity of the Pickup and Delivery Problem with Split Loads, and is a natural extension to the Split Delivery Vehicle Routing Problem (SDVRP) since, as mentioned before, the Vehicle Routing Problem is a special case of the Pickup and Delivery Problem where all the delivery locations are the depot.

THEOREM 1 (From Theorem 1 in [6]). *Given the pickup and delivery locations of a set of k loads, a set of m vehicles of capacity Q, and a very small value ϵ, let v(PDPSL) be the cost of the optimal PDPSL solution to deliver these loads, and let v(PDP) be the cost of the optimal PDP solution. Then the ratio v(PDP)/v(PDPSL) is maximized when the loads are all of size $Q/2+\epsilon$, as $k \to \infty$.*

Proof: Given a fixed number of loads and a set of origin and destination locations, the largest reduction in cost through the use of split loads occurs when the ratio of loads that must be delivered individually in a PDP solution relative to a PDPSL solution is maximized. As this ratio increases, fewer loads may be placed on the vehicle concurrently for delivery with the PDP solution relative to the PDPSL solution. With less opportunity to delivery loads simultaneously, *v(PDP)* increases relative to *v(PDPSL)*. Therefore, the load size that maximizes the ratio of loads that must be delivered individually in a PDP solution relative to a PDPSL solution is also the load size that maximizes the ratio *v(PDP)/v(PDPSL)*.

Consider any positive integer $m \geq 2$. If all load sizes are $Q/m + \epsilon$, we can combine $m - 1$ loads without splitting or m loads with splitting (after removing each ϵ). Therefore, the ratio of the number of loads that must be delivered individually between the PDP and PDPSL cases will be $\lceil k/(m-1) \rceil / (\lceil k/m \rceil + 1)$. As m increases (i.e., as load sizes decrease) and $k \to \infty$, this ratio decreases.

When $m = 2$, the ratio of the number of loads that must be delivered individually between the non-split and split cases become $k/(\lceil k/2 \rceil + k\epsilon)$. This ratio decreases with an increase in ϵ (especially when $\epsilon > Q/k$, the ϵ portions of the loads can no longer be combined onto one vehicle). Therefore, the ratio of loads that can be delivered concurrently in a PDP solution relative to a PDPSL solution is maximized when load sizes are equal to $Q/2 + \epsilon$.

CONJECTURE 2 (From Conjecture 2 in [6]). *Given any set of loads requiring service, assume that v(PDPSL) is the cost of the optimal PDPSL solution to deliver these loads and v(PDP) is the cost of the optimal PDP solution. Then,* $v(PDP) \leq 2v(PDPSL)$.

In the example in Section ch1overview, the maximum ratio of costs was found to be two. Although a bound of two may be shown to exist for specific simple cases of the PDPSL (e.g., when an optimal PDPSL routehas at most one

split load on the vehicle at any time), a general bound is more difficult to define. In [6], they found no instance where all loads in an optimal PDP solution can be delivered complete with a cost that is greater than double that of the PDPSL cost and the property is raised as a conjecture. In out numerical testing, we also found no instance that violate this property. However, we can not prove this property for every instance of the problem either.

THEOREM 3. *The PDPSL is NP-hard.*

It is shown in [14] that the Skip Delivery Problem, in which each of the vehicles has a maximum capacity of two skips, is NP-hard. This is a special case of the general Split Delivery Vehicle Routing Problem. While the general Split Delivery Vehicle Routing Problem is a special case of the Pickup and Delivery Problem with Split Loads where all the delivery locations are the depot, so the Pickup and Delivery Problem with Split Loads is also NP-hard.

Chapter 3

Solving the Pickup and Delivery Problem with Split Loads

Metaheuristics and local search are prevalently used nowadays in solving difficult computational problems, especially in the family of Vehicle Routing Problems. As the VRP in its general form is a special case of the Pickup and Delivery Problem in its general form, where all the delivery locations are the depot, metaheuristics and local search are also applicable to the PDP. When split loads are permitted in the PDP, the solution space is enlarged rapidly since each of the loads could be a candidate for one or multiple splits and each of the splits can be of whatever quantity as long as it is less than the residual amount of the load. Due to this reason, we cannot expect a perfect exact solution to the PDPSL to get its optima when the number of loads are large (in small cases, dynamic programming could be used to get an optimal solution, see [7], but even in this situation, the task is time consuming), instead, we would rather develop efficient and effect metaheuristic and local search schemes so we are able to solve practical sized problems.

In the dissertation, we propose a tabu-embedded simulated annealing-like algorithm which restarts a search procedure from the current best solution after several non-improvement search iterations. The idea of this hybrid metaheuristic is from [11]. The difference of this algorithm with the traditional implementation of the simulated annealing algorithms is that instead of performing the simulated

annealing procedure on the probabilistically accepted solution repeatedly until the procedure terminates, this algorithm compels the simulated annealing procedure to restart from the current best solution after several simulated annealing iterations without any improvement, and the algorithm terminates after a predetermined number of restarts without improvement. Our algorithm is different from theirs, too, in the first place, we are not using standard simulated annealing procedure to probabilistically accept a new solution because we have many sub-processes in the procedure, while the creating-split-load and inserting-load processes may increase or decrease cost, we are always seeking cost reduction in other processes like combining-routes, exchanging-loads, rearranging-loads. So, we adopt the probabilistically accepting policy in creating-split-load and use a predetermined number of iterations, instead of a random number of iterations, to control the simulated annealing procedure; in the second place, we are solving the PDPSL problem and they were solving the Pickup and Delivery Problem with Time Windows so we have not only the load-inserting, load-exchanging, load-rearranging operators, but also load-splitting operator to create split loads.

In [6], a similar tabu search algorithm, with predetermined number of iterations and restarting scheme from the current best solution, was proposed to solve the PDPSL as well, but their algorithm is more complicated than our. The differences between their algorithm and ours lie in the following aspects: (1) they have separate tabu lists for the first and second split on a load and the maximum allowed splits on a load is 2 in their algorithm; while in ours, we first investigate the effect of the number of load splits, i.e. compare the performance when different number of load splits are applied, and conclude that the number of load splits does not affect much the cost reduction (see our first numerical test in Chapter 4); based on this conclusion, we use a simpler tabu structure for the split loads which does not distinguish whether the split load is the first or the second or the third

split on a certain load, and we allow a maximum of 3 splits for the subsequent numerical tests in Chapter 4; (2) in their algorithm, there are two levels of inner loops and at the first level, the load splitting process is called to create a possible split load, the first level of inner loop is controlled by a predetermined fix integer and the iterator is keeping increasing; at the second level, combining route, load exchanging and load inserting processes are called to optimize the current solution generated by the load splitting process, and the second level of inner loop is controlled by a predetermined fixed integer and the iterator is set to be 0 when the current solution is optimized. While in our algorithm, the load splitting operator is also put in the second inner loop and the second inner loop is controlled in the same way, but we reset the iterator of the first inner loop to be 0 if the second inner loop comes out with a better solution, thus resemble the way used in [11].

The chapter is organized in this way that in section 3.1, we introduce the way how we create an inial solution to the problem; in section 3.2, we develop a series of operators, i.e. load splitting operator, load insertion operator, load exchanging operator, load rearranging operator, which enable us to explore the neighborhood of a given solution, specifically, in section 3.2.1, we introduce how we create a split load and the criteria to accept the split load; in section 3.2.2, we introduce how a load can be inserted from a route to another route or a place on a route to another place on the same route, and what will be performed on the associated pickup and delivery locations; in section 3.2.3, we describe how a pair of loads is exchanged, whether from two different routes or a same route, and what will be performed on the associated pickup and delivery locations, naturally, a load exchanging operator is a combination of two load insertion operator on two different loads; in section 3.2.4, we describe how we rearrange the pickup and delivery locations on a route to get a better solution.

In developing the metaheuristics, we assume that the amount of each load is

less than the vehicle capacity and there is only 1 vehicle. We will briefly introduce how to deal with the case when the amount of some loads is greater than the vehicle capacity (i.e. loads have to be split) and show some numerical results, and we will also briefly discuss the situation when multiple vehicles are available in Chapter 4.

3.1 Constructing Initial Solution

In constructing the initial solution, we do not consider splitting load but just transport the load from the pickup location to the corresponding delivery location as a whole. There are two prevalent ways to construct initial solutions to VRP and PDP problems, one way is to create a route for each load and eliminate some of the routes in the subsequent processes; the other is to insert the selected load into the existing routes using the insertion heuristic in [12], and during the insertion procedure, the selected load is inserted into the best suitable location in the partially-constructed route that cause the minimal increment in travel cost. After all the loads are inserted, an initial solution is obtained. There is no difference in the 1 vehicle case and we follow the first way in the algorithm.

3.2 Neighborhood Structures

Load swap operators are commonly used in the literature to explore neighborhood in local search procedures, as the solutions to these kinds of problems are often naturally represented by some routes with sequences of customer nodes which are visited by the vehicles to provide the desired service. We use three load swap operators in performing local search within the algorithm, which are very prevalent in previous researches, and they are load insertion operator, load exchanging operator, and load rearranging operator. Furthermore, to explore the even larger neighborhood space under the condition that a load can be transported

by many vehicles or many times by the same vehicle, thus only a portion of the load might be delivered each time, we develop a load splitting operator which moves a portion of a load on one route to another route, or to a different segment on the same route, to fill up the excess capacity in that route segment. We use three operators, load split operator, load insertion operator, and load exchange operator, to generate neighborhood candidates for the metaheuristic procedure, and we use the load rearranging operator to do post-optimization within the other operators.

3.2.1 Load Splitting Operator

The load split operator moves a portion of a load on one route to a different segment on the same route, or a segment on another route, to fill up the excess capacity in this segment, subject to all the constraints imposed on PDPTWSL. In generating the split load, a load is randomly selected and its size is compared to all occurrences of excess capacity along the same route, or on another route. If the size of the load is greater than one of these occurrences of excess capacity, the load might be split, with the portion to be moved that has the same size as the excess capacity at the location on the route it is moved to. That is, if a load of size 0.8 loads is to be split and there are 0.5 loads of excess capacity elsewhere on the same route or on another route, 0.5 loads will be moved to fill up that excess capacity, and 0.3 loads will remains in the original position on the route.

Figure 3.1 depicts the split load creation, where the route segment 1 visits $P1$, deliveries load 1 from $P1$ to $D1$, and has the excess capacity of 0.5 to be filled in, thus load 2 could be split, with 0.5 loads to be delivered with load 1, and the left 0.3 load to be delivered in the original route segment. The final sequence of the locations P1, P2, D2, D1 may not be as arranged in the figure since load rearranging operator may change this order.

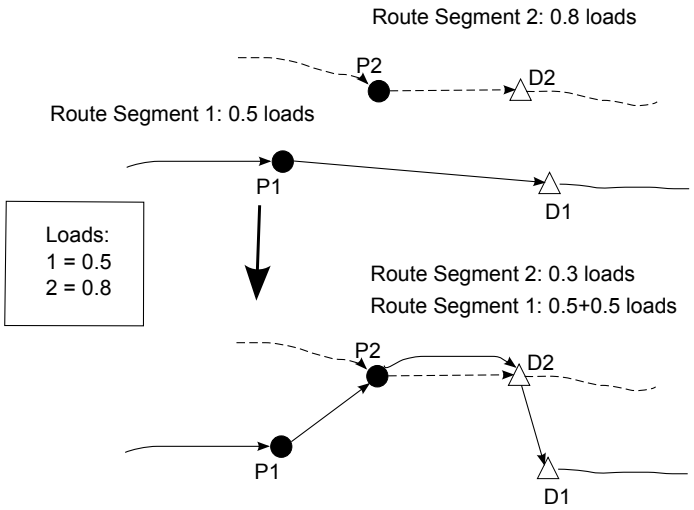


Figure 3.1: Load Split Operator

3.2.1.1 Criteria to Accept a Split

Creating a split load incurs an increase in the total travel cost and can never reduce the cost. Following the method used in [7], we accept a split load if the ratio of the incurred cost times 8 to the original cost of the route is less than a randomly generated decimal number that is between 0 and 1, and if the ratio is greater than 0.8, it is capped at 0.8. The parameter is set to get best possible result after several tests.

3.2.2 Load Insertion Operator

The load insertion operator moves one load, consisting of a pickup location and a delivery location, from one route to another route, or from one route segment to another route segment within the a single route.

In Figure 3.2, P1 and D1 are originally two locations in route segment 1. The load insertion operator first removes the two locations from the route where route segment 1 resides, and then insert them to optimal positions in the route where route segment 2 resides, subject to P1 being visited before D1. The final sequence

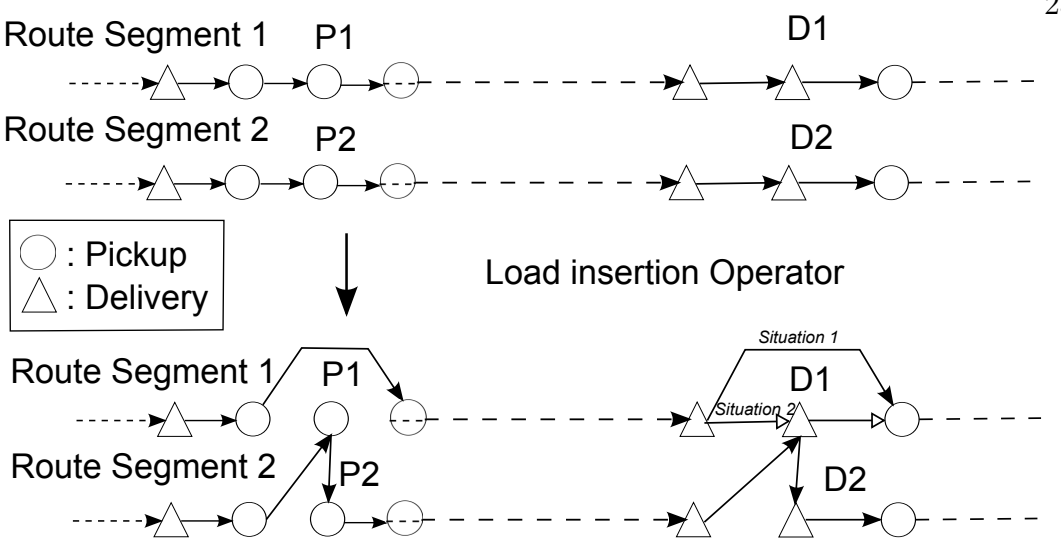


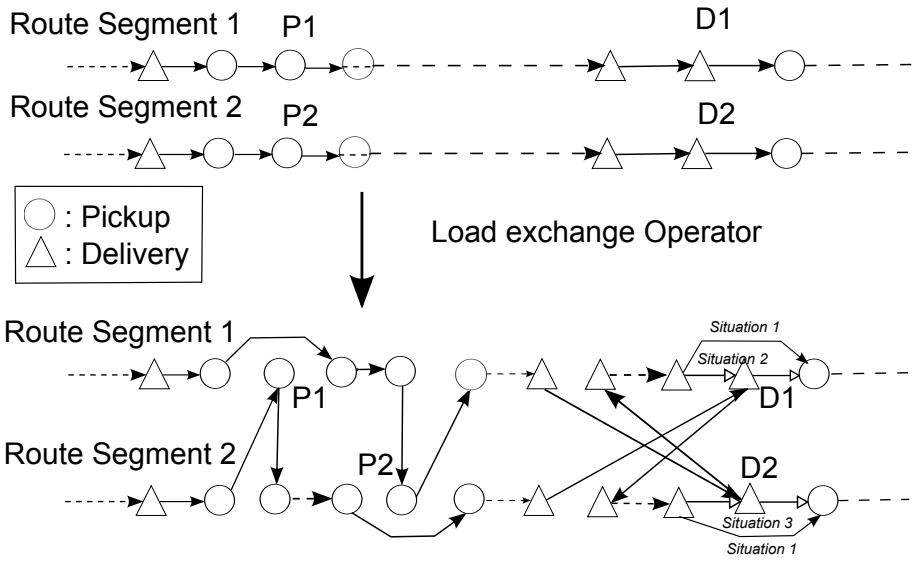
Figure 3.2: Load Insertion Operator

of the locations P1, P2, D2, D1 in route segment 2 may not be as arranged in the figure since load rearranging operator may change the order. As split loads are allowed, there are two cases that deserve attention regarding removing the delivery location D1:

- If there are other loads, whether from the same load that is picked up at P1 and is to be delivered to D1, or from another load that is picked up somewhere else and is to be delivered to D1, on the route segment that are to be delivered to D1, don't remove D1;
- If there are no other loads on the route segment that are to be delivered to D1, remove D1.

3.2.3 Load Exchanging Operator

The load exchanging operator exchanges one load, consisting of a pickup location and a delivery location, with another load on another route, or in another route segment within the same route.



***Situation 1**: if there are no other loads on the route that RS1 belongs to which are to be delivered to D1, or there are no other loads on the route that RS2 belongs to which are to be delivered to D2;*
***Situation 2**: if there are other loads on the route that RS1 belogns to which are to be delivered to D1;*
***Situation 3**: if there are other loads on the route that RS2 belongs to which are to be delivered to D2*

Figure 3.3: Load Exchanging Operator

The load exchanging operator is actually a combination of two load insertion operators. In Figure 3.3, P1 and D1 are originally two locations in route segment 1 and P2 and D2 are originally two locations in route segment 2. The load exchanging operator first remove P1 and D1 from the route that route segment 1 resides (route 1), and P2 and D2 from the route that route segment 2 resides (route 2), and insert P1 and D1 into route 1 optimally, insert P2 and D2 into route 2 optimally. Based on the same consideration as that in the load insertion operator, we need to be prudent when removing the delivery locations D1 and D2 from its respective original route.

3.2.4 Load Rearranging Operator

Within the same route, the load rearrange operator rearrange the pickup and delivery locations of a load to the best positions that maximally reduce the travel cost.

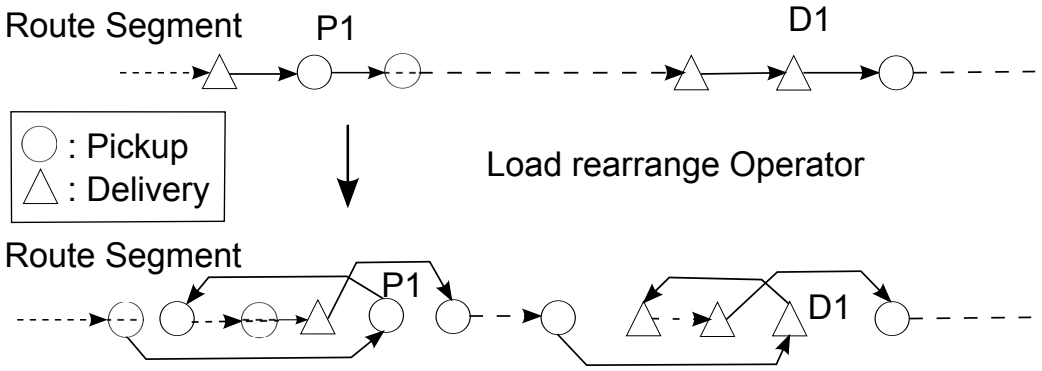


Figure 3.4: Load Rearranging Operator

In Figure 3.4, P1 and D1 are pickup and delivery locations of a load in a route. The load rearranging operator first removes P1 and D1 from the route then inserts them into the best positions within the same route.

3.3 Metaheuristics

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

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

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

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

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

$WORKCOST$ - 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 iter1 and iter2 equal to 0.

While ($iter1 < ITERMAIN1$)

{

Set $WORKCOST$ equal to $BASECOST$;

Set $TEMPCOST$ equal to $WORKCOST$;

While ($iter2 < ITERMAIN2$)

{

Create Split Loads. Update $TEMPCOST$.

Combine Routes. Update $TEMPCOST$.

Intra-Route Load Exchange. Update $TEMPCOST$.

Intra-Route Load Insertion. Update $TEMPCOST$.

Inter-Route Load Exchange. Update $TEMPCOST$.

Inter-Route Load Insertion. Update $TEMPCOST$.

If $TEMPCOST < WORKCOST$, set $WORKCOST = TEMPCOST$, and $iter2 = 0$.

Else increment $iter2$;

}

If $WORKCOST < BASECOST$, set $BASECOST = WORKCOST$, and $iter1 = 0$.

Else increment $iter1$;

}

3.3.1 Create Split Loads

The following additional lists and variables are used for the algorithm to create split loads:

insertdone - the list of loads that have been previously inserted into a route using the 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.

itersplit - current number of iterations for split load creation.

$ITERINNERSPLIT$ - 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 ($itersplit < ITERINNERSPLIT$) {

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 *insertdone* or *tabusplit* or $tabusplitfail$.

Calculate cost of generating split load, $SPLITCOST$.

Accept split load if $SPLITCOST * 8/TEMPCOST < RANDNUM$, where $RANDNUM$ is a randomly generated decimal number between 0 and 1.

If split load creates violation of number of split loads, add to $tabusplitfail$, and iterate *itersplit*.

If split load valid, add to *tabusplit* and update cost, such that $TEMPCOST =$

$TEMPCOST + SPLITCOST$. EXIT the procedure.

Increment *itemsplit*.

}

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*.

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

Continue until no route combination results in cost improvement.

3.3.3 Descent Local Search

As an extension of local search, descent local search (DLS) starts from an initial solution. All the solutions in the neighborhoods are checked. The best

solution with the minimum objective cost will act as the initial solution for repeating this DLS procedure and the search is repeated until no improvement is found. The DLS procedure operates as follows:

Input: A solution S; **Output**: A local optimal solution S_b

1). $S_b \leftarrow S$

2). Select the best solution $S_b^{'}$ with minimum traveling cost within a defined neighborhood of S_b.

3). **IF** $S_b^{'}$ is better than S_b, **THEN** $S_b \leftarrow S_b^{'}$; **GOTO** Step 2.

4). **ELSE RETURN** S_b.

The Intra-Route Load Exchange, Intra-Route Load Insertion, Inter-Route Load Exchange, Intra-Route Load Insertion procedures are actually Descent Local Search procedures and are described in the sections below.

3.3.4 Intra-Route Load Exchange

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, $SWAPCOST$. and reject those that are on $swap1fail$, $swap1done$ or if swap violates vehicle capacity.

Select swap with minimum $SWAPCOST$, if any $SWAPCOST$ 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 swap is valid, add to *swapdone* and update cost such that $TEMPCOST = TEMPCOST + SWAPCOST$. Reset $swap1fail$.

Continue until no load swap results in cost improvement.

The **Inter-Route Load Exchange** procedure resembles the one described here except the exchange is done between two routes.

3.3.5 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, $INSERTCOST$.

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

Select insertion with minimum $INSERTCOST$, if any $INSERTCOST$ 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 insertion is valid, add to $insert1done$ and update cost such that $TEMPCOST = TEMPCOST + INSERT1COST$. Reset $insert1fail$.

Continue until no insertion results in cost improvement.

The **Inter-Route Load Insertion** procedure resembles the one described here except the insertion is done between two routes.

3.4 Conclusion

In this chapter, a metaheuristic for solving the PDPSL is presented, which is a tabu-embedded simulated annealing like algorithm which restarts a search procedure from the current best solution after several non-improvement search iterations. This heuristic combines elements from several methods such as load splitting, route combination, load insertions, load exchanging and load rearranging. The method for generating split loads and the criteria to accept the split loads are described, along with the various algorithms used in the metaheuristic. And the metaheuristic framework is also presented.

Chapter 4

Numerical Tests

In [6], the authors have claimed that some benefit can be found for almost all the problems that are randomly generated, with certain instances displaying a cost reduction of more than 30% with the use of split loads, and they have also found from the average outcomes of the testing cases that the most benefit is found when load sizes are just over one half of vehicle capacity.

In this section, we want to numerically test the new metaheuristic proposed in the Chapter 3 with large-scale data. As there is no benchmark for the Pickup and Delivery Problems, we follow [6]'s way and generate the set of problems. The authors in [7] and [6] have used the heuristics that they developed to address the following questions:

(1) What is the benefit of cost reduction if split loads are allowed?

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

In addition, we want to address the following question:

(3) Is there huge difference due to the maximum allowed number of split loads if that is not limited?

In Section 4.1 the design of the experiments used to evaluate split loads is described, and in Section 4.2 the results of the experiment are presented. Conclusions are drawn in the last section.

4.1 Experimental Design

We generated problem sets of three sizes, with 75, 100, or 125 transportation requests, which are similar to those used in testing the SDVRP ([3], [4]). Each transportation request contains the origin and destination location information, and the amount 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, and the depot is 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 origin locations from where a load is picked up. So the 75 request problem has 15 destination locations, the 100 request problem has 20 destination locations, and the 125 request problem has 25 destination locations. As is reported in [6], testing with a different number of origins found no meaningful changes to the results, we also find the same results. Every origin-destination combination has a load to be delivered between the two locations, such that we generated exactly 75 requests, 100 requests and 125 requests for these three problem sets respectively. Three configurations with different origin and destination locations were generated for each problem size.

The load sizes were also randomly generated. In this chapter, 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 and 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 smaller 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]. In [6], load sizes below 0.1 or above 0.9 were not used because they found splits of these loads rarely occur and any benefit was negligible. We did not use these two load sizes either, because instead of finding negligible cost reduction, we found that splitting load under both situations lead to cost increase in most of the time. 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 five 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 five load configurations, resulting in 15 different problems for each load range, and 60 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 until it stops.

In [7], it is shown that one split per route is not optimal for the PDPSL, although it has been proven for the Split Delivery Vehicle Routing Problem [13], we test additional scenarios where the number of splits on a load (not a route) is varied from 2 to 6 to see whether the performance is greatly affect by this alteration. For each number of maximum allowed splits, we generated two different configurations of pickup and delivery locations with X, Y coordinates drawn from a uniform distribution over the same range, and five different load configurations with each load drawn from a uniform distribution over the load range [0.51, 0.6] because the most benefit is found within this load range and the metaheuristic is more stable given this load range configuration, thus ten different problems for each number of allowed splits and 60 problems in all.

The metaheuristic was coded in C++ and all experiments were run on a 1.5 GHz Celeron M processor with 1 GB RAM under Windows XP Service Pack 3.

4.2 Experimental Results

We report the numerical test results on different maximum number of allowed splits in Section 4.2.1, and we present results on problem sets with smaller load ranges in Section 4.2.2, and finally results on problem sets with wider load ranges are presented in Section 4.2.3.

4.2.1 Performance with Different Maximum Allowed Number of Splits

Figure 4.1 presents the average percentage reduction in cost when split loads are allowed for the 75-, 100-, and 125-request problem sets with loads generated over the load range [0.51, 0.6], and the maximum allowed number of splits is from 2 to 6. The costs are based on the distance traveled by the vehicle.

The cost reductions seem to be consistent, with a maximum cost reduction of

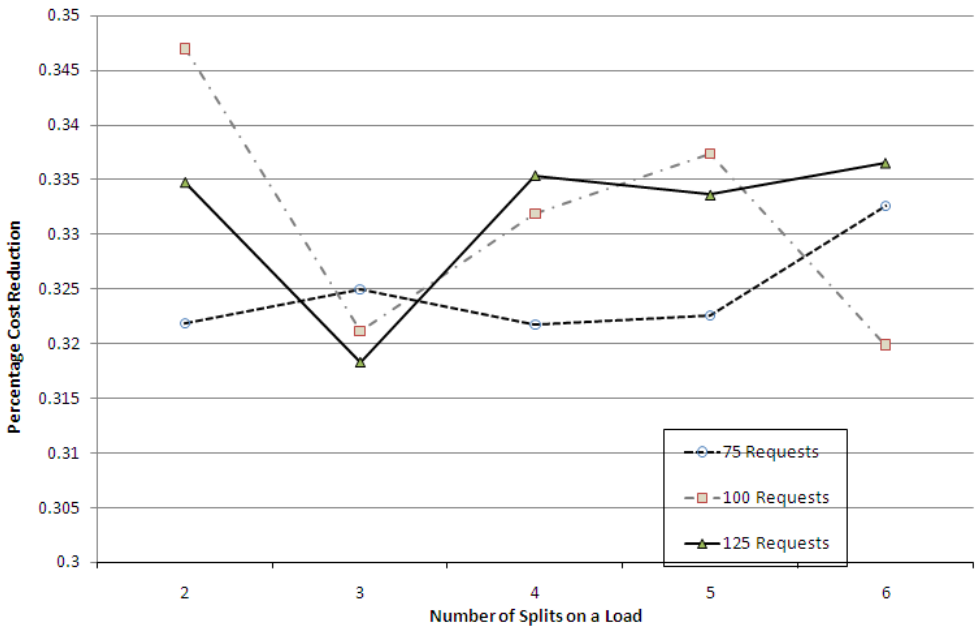


Figure 4.1: Average Percentage Cost Reduction with Split Loads for Different Maximum Allowed Number of Splits for Load Range [0.51, 0.6]

nearly 35% for the 100-request problems and when the maximum allowed number of splits is 2, and a minimum cost reduction of nearly 32% for the 125-request problems and when the maximum allowed number of splits is 3. There is also no huge difference in cost reductions for each of the problem sets, which is consistent with the result in [6], where they reported almost the same cost reductions (around 28%) for the 75-, 100-, and 125-request problem sets when the loads are within the load range [0.51, 0.6]. Our algorithm seemed to be more effective in fostering split loads within this load ranges so that greater cost reduction is possible. (See also in the Appendix for a comparison of the average number of splits created by the algorithm in [6] and by our algorithm.)

4.2.2 Performance on Smaller Load Size Ranges

Figure 4.2 presents the average percentage cost reduction when split loads are allowed for the 75-, 100-, and 125-request problem sets with smaller load size

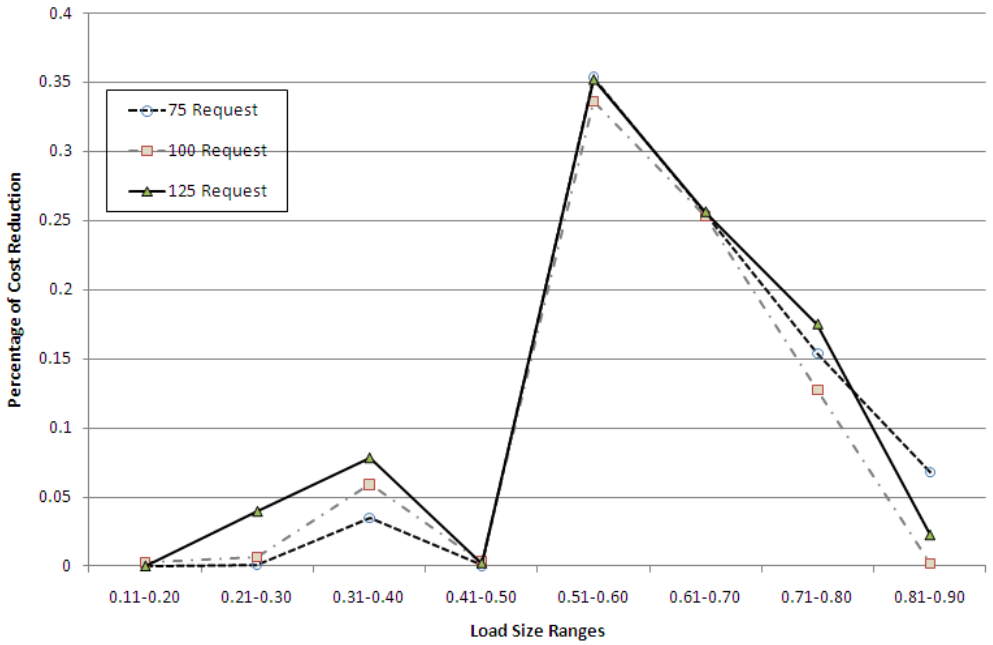


Figure 4.2: Average Percentage Cost Reduction with Split Loads for Each Load Range Tested for the 75-, 100-, and 125-Request Problem Sets

ranges. Different from the results in [6], we found no benefits for the loads in the range [0.11, 0.20], and found relatively greater cost reduction for the load size ranges that are greater than [0.51, 0.6] (including [0.51, 0.6] as well). By comparing the average number of split loads created by out algorithm and by the the algorithm in [6], we found an huge increase in this number and it was due to the reset scheme of the iterator in the inner loop in our algorithm, so that exploring even larger solution space is possible.

Even though some cost reduction is found with the use of split loads for almost every load range, more benefit is clearly found with certain load sizes. The results for each problem set support a main theoretical result in [6] and also in previous chapter. 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 reducation 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 benet 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.

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 20% for the range [0.71, 0.8] and even less than 10% for the range [0.81, 0.9].

Table 4.1 presents the average CPU time, in seconds, required to find the solutions reported and the average number of split loads of the solution. The counter for the iterations is incremented each time the heuristic begins the split load creation step, prior to beginning the load combination sub-algorithm (and continuing on to the other local improvement sub-algorithms), and the time for updating temporary data and outputting the results is omitted.

The CPU time increases quickly with problem size for every load range, doubling with an increase of 25 requests in most situations. The increase in the

Table 4.1: Average CPU Time, No. of Splits and Cost Reduction for Problem Sets with Split Loads on Smaller Load Size Ranges

	Problem Size								
	75			100			125		
Load Range	CPU Time	Splits	Cost Reduction	Time	Splits	Cost Reduction	Time	Splits	Cost Reduction
0.11-0.20	4.0007	0	0	8.791	0.7333	0.00297	13.795	0	0
0.21-0.30	3.6	1.8667	0.00118	9.3826	3.1333	0.00655	24.963	11.0667	0.0395
0.31-0.40	6.6566	13.0667	0.03484	20.257	23.8	0.05901	40.973	34.467	0.0783
0.41-0.50	4.934	3.6	0.00054	6.504	1.667	0.00324	14.542	5.0667	0.0022
0.51-0.60	22.3327	48.0667	0.35354	48.546	61.6	0.33619	106.818	81.533	0.3519
0.61-0.70	18.0363	42.93	0.255	35.923	58.333	0.25312	59.978	66.867	0.2563
0.71-0.80	32.3223	61.467	0.154	55.923	69.667	0.1277	141.13	101.7333	0.175
0.81-0.90	19.7783	37.733	0.068	6.88	2.6667	0.00195	26.7741	14.7333	0.0225

computational time is mainly attributed to the greater number of steps that must be performed within each sub-algorithm to find an improvement. As problem size increases, the number of combinations that can be tested for local improvement is increased significantly and rapidly.

4.2.3 Performance on Wider Load Size Ranges

Table 4.2 presents the average percentage cost reduction when split loads are allowed for the 75-, 100-, and 125-request problem sets with wider load size ranges. Again, as with the smaller 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 remarkable as for some of the smaller ranges, the average cost reduction for the 100-request with loads in range [0.5, 1.0] was found to be more than 21%. Also, the results regarding the most beneficial load sizes support those found with the smaller load size ranges. The range [0.5, 1.0] finds the most benefit in every situation. Also, the ranges [0.3, 0.7] and [0.1, 1.0] are comparable in terms of cost benefit, but slightly lower than the range [0.5, 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, while this load variation can greatly deteriorate the benefit by the use of split loads. The range [0.1, 0.5] finds the least benefit, as the smaller loads are less likely to be split. And for the average CPU times on these wider load size ranges, they present the same pattern as that in the smaller range tests that it increases quickly with problem size for every load range, doubling with an increase of 25 requests in most situations.

Table 4.2: Average CPU Time, No. of Splits and Cost Reduction for Problem Sets with Split Loads on Wider Load Size Ranges

	Problem Size								
	75			100			125		
Load Range	CPU Time	Splits	Cost Reduction	Time	Splits	Cost Reduction	Time	Splits	Cost Reduction
0.1-1.0	14.736	18.933	0.0541	24.98	25.4	0.0471	40.005	26.2667	0.0417
0.1-0.5	10.618	10.125	0.0367	15.849	13.2	0.0309	21.456	11.333	0.0309
0.5-1.0	18.913	40.067	0.1859	48.016	61.533	0.2138	82.466	80	0.1861
0.3-0.7	12.247	16.6	0.0624	26.681	28.4	0.0694	28.631	26.4	0.0726

4.3 Conclusion

We tested the metaheuristics developed in the previous Chapter on randomly generated data sets with load sizes falling in several ranges and quantified the benefits of using split loads. We first investigated the relationship between the maximum allowed number of splits on a load and the cost reduction by the use of split loads, and found that the benefits from using the split loads vary within a small range when the maximum number of allowed split loads increases from 2 to 6, and the trend is predicted to be consistent. By limiting the maximum allowed splits on a load to be 3, we found that the most benefit to be with loads just over half of the vehicle capacity for the smaller load ranges, and this most benefit was around 35% cost reduction for all the three problem sets, which was 5% more cost reduction than that found in [6]. For the wider load ranges, the most benefit is found with loads ranging from 0.5 to 1.0, and this most benefit was around 20%, which was also a significant increase compared with the results reported in [6]. One plausible explanation for this phenomena is that our algorithm is tabu-embedded simulated annealing like metaheuristic, and both the inner and outside loops are compelled to reset the iterator to be 0 if some improvement is found, while their algorithm is a simple combination of many heuristics, so that exploring even larger solution spaces is possible in our algorithm.

Chapter 5

Extension to The Basic Problem

In this Chapter, we consider the case where there are some (or all the) loads that are greater than the vehicle capacity , through numerical tests. In this situation, the loads that are greater than the vehicle capacity must be split. A common method in the literature is to split the loads in a way that the vehicle is engaged for each of the loads for multiple time, with the first few times filling the whole vehicle capacity and the last time carrying the residual amount of the load that is being transported, and then take these multiple times of transporting a same load as transporting multiple different loads, and optimize the routes (Common Split hereafter). As illustrated in Chapter 3 and Chapter 4, most benefit through splitting loads is available when all the loads are just a little more than half of the vehicle capacity. This conclusion prompts us to re-think about the common way to split loads when some/all the loads are greater than the vehicle capacity. If we split the loads that are greater than the vehicle capacity in a way that not full vehicle capacity amount of load is continually transported until the amount of load left is less than the vehicle capacity, but other quantity of the load (less than the vehicle capacity, say, 0.85 of the vehicle capacity) is continually transported until the amount of load left is less than this quantity, what is the result? Obviously we are increasing the number of "loads" to be carried compared with the Common Split way, thus making the transportation network even more

complicated; however, the more complicated network may foster the aggregation of loads in close proximity and increase the odds that more loads can be split and re-combined together in a better way, thus reducing the cost back to its level in the Common Split way, and maybe even less than that level.

We investigate this trade-off between splitting greater-than-vehicle-capacity loads with different quantity and fostering the cost-reducing effect by the use of splitting loads through numerical tests in this Chapter, and specifically, we have two questions in mind to be investigated:

(1) Whether the Common Split way is the best?

(2) What is the difference if we split the loads with different quantities?

In Section 5.1, we report the experimental design for this extension to the basic problem, and in Section 5.2, we tabulate main findings from the numerical study, conclusion comes in the last section.

5.1 Experimental Design

We adopt the same configuration as that used in Chapter 4, except the loads are randomly generated within different load size ranges. The third load size range grouping contains three different load size ranges, namely, [1.0-2.0], [0.5-1.5], [0.1-2.0]. So, each of the three location configurations is matched with each of the five load sets within a load range, resulting in 15 different problem instances for each load range and 45 instances overall.

For each problem instance, we adopt the Common Split way as well as five less-than-vehicle-capacity split policies, which split the loads that are greater than the vehicle capacity in different amount of load. For example, a 0.55-Split policy is to split a load that is greater than the vehicle capacity in a way that 0.55 amount

of the load is transported each time the vehicle visits the origin of the load until the residual amount is less than 0.55, so that the residual amount can be finally transported. These five policies are 0.55-Split, 0.65-Split, 0.75-Split, 0.85-Split, and 0.95-Split.

5.2 Experimental Results

Table 5.1 presents the average CPU time, in seconds, required to find the solutions reported, using respective split policy, and cost reduction by using split loads (compared to the case when split loads are not allowed). The counter for the iterations is incremented each time the heuristic begins the split load creation step, prior to beginning the load combination sub-algorithm (and continuing on to the other local improvement sub-algorithms), and the time for updating temporary data and outputting the results is omitted.

In general, the CPU time increases if the loads are split in less-than-vehicle-capacity amount, instead of adopting the Common Split polity, sometimes, the increment is very significant, reflecting the increased complexity of the transportation network; however, the performance seems to remain the same for all the three load size ranges, and the cost reduction is neglectable, which means on the one hand, although there is some benefit by the use of split loads when some of the loads to be transported are greater than the vehicle capacity, this benefit is neglectable in some sense and to achieve this benefit is time consuming; on the other hand, there seems to be no optimal split policy that can always guarantee the best quality of solutions.

Table 5.1: Average CPU Time and Cost Reduction for Problem Sets with Split Loads on Different Split Policies

		75		100		125	
Load Range	Split Policy	Cost Reduction	Time	Cost Reduction	Time	Cost Reduction	Time
1.0-2.0	1	0	13.80098333	0	31.5538	0	75.00325
	0.95	-0.032175821	50.496	-0.025403912	163.7868	-0.01027843	110.5285
	0.85	-0.03204762	31.23163333	-0.030108711	89.88035	0.02575148	499.0195
	0.75	-0.021496156	64.25141667	-0.048649908	89.41575	0.00999235	333.4035
	0.65	-0.017449181	92.3946	-0.03042199	120.62155	0.000263561	112.4245
	0.55	-0.024928168	43.2193	-0.030220033	151.8599	0.009779743	205.036
0.1-2	1	0.016052369	20.98963333	0.01670502	46.57765	0.01852385	416.715
	0.95	0.008355518	39.48436667	0.011786075	43.0378	0.005019017	875.6648333
	0.85	-0.006686486	36.97793333	-0.006185366	46.1452	-0.003579925	380.42155
	0.75	-0.001617915	31.248	0.002340652	19.98965	0.003282352	505.9948333
	0.65	0.014143806	35.99643333	-0.014018136	26.1043	-0.008875788	218.87305
	0.55	-0.0113229	8.446213333	0.00919525	33.95845	0.006545791	561.7134333
0.5-1.5	1	0.004813111	10.173342	0.027581382	50.10294	0.016036644	79.86768
	0.95	0.021026924	29.05402	0.029649724	85.80986	-0.021399114	141.14312
	0.85	0.023560455	21.532908	0.024511666	97.97718	-0.016869542	118.99012
	0.75	0.012655126	14.509996	0.006658798	55.40736	-0.026405599	148.2464
	0.65	0.000934579	23.715168	0.020820987	55.89308	-0.016614787	125.88772
	0.55	0.011743381	30.37056	0.037274717	79.30248	-0.006762859	133.66824

5.3 Conclusion

In this section, we investigate through numerical study the extension to the basic problem, where there is some possibility that the amount of some of the loads is greater than the vehicle capacity. We find that in general there are some benefits by applying split loads to the transportation network design even when the loads are general and those loads that are greater than the vehicle capacity have to be split, and we also find there is no optimal splitting policy that can always guarantee the best quality of solutions.

Chapter 6

Conclusions

In this dissertation, we studied the use of split loads for the Pickup and Delivery Problem, and proposed a metaheuristic with annealing like restart strategy to guide the local search in three neighborhoods that were defined to solve the problem, and the metaheuristic was based on a predetermined fixed number of restarts annealing like procedure with tabu-lists to avoid cycling in the search process. We tested the algorithm on several sets of problem instances generated with different transportation requests and over different load size ranges. The experimental results on these problem sets have shown that benefits are common if split loads are adopted in designing practical sized transportation network for different load size configurations, and the most benefit is seen when all the loads are just a little above half of the vehicle capacity and have small variations. Besides the basic problem, we also investigated an extension. We found that in general there are some benefits by applying split loads to the transportation network design even when the loads are general and those loads that are greater than the vehicle capacity have to be split, and we also found there is no optimal splitting policy that can always guarantee the best quality of solutions.

Bibliography

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

[2] Dror, M., G. Laporte, P. Trudeau. 1994. Vehicle routing with split deliveries. Discrete Appl. Math. **50**, 239-254.

[3] Dror, M., P. Trudeau. 1989. Savings by split delivery routing. Transportation Science **23**, 141-145.

[4] Dror M, P. Trudeau. 1990. Split delivery routing. Naval Research Logistics **37**(3), 383-402.

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

[6] M. Nowak, Ö. Ergun, C. C. White 2008. Pickup and delivery with split loads. Transportation Science **42**(1), 32-43.

[7] M. Nowak 2005. The Pickup and Delivery Problem with Split Loads. Ph.D Thesis. Industrial and Systems engineering, Georgia Institute of Technology.

[8] Yang, J., P. Jaillet, H. Mahmassani. 2004. Real-time multivehicle truckload pickup and delivery problems. Transportation Science **38** 135-148.

[9] Psaraftis, H. N. 1988. Dynamic vehicle routing problems. B. L. Golden, A. A. Assad, eds. Vehicle Routing: Methods and Studies. Elsevier (North-Holland), Amsterdam, 223-248.

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

[11] Li, H., A. Lim. 2001. A metaheuristic for the pickup and delivery problem with time windows. A. K. Bansal, ed. 13th IEEE Internat. Conf. Tools with Artificial Intelligence (ICTAI), Dallas, 160-170.

[12] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. 1987. Operations Research **35**, 254-264.

[13] 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.

[14] C. Archetti, R. Mansini, M. G. Speranza. 2005. Complexity and Reducibility of the Skip Delivery Problem. Transportation Science **39** 182-187.

[15] Jeffrey W. Ohlmanm, Michael J. Fly, Barrett W. Thomas 2008. Route Design for Lean Production Systems. Transportation Science. **42**, 352-370.

[16] Belenguer, J. M., M. C. Martinez, E. Mota. 2000. A lower bound for the split delivery vehicle routing problem. Operations Research **48**, 801-810.

[17] Archetti, C., A. Hertz, M. G. Speranza. 2006. A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science **40**, 64-73.

[18] Archetti, C., M. W. P. Savelsbergh, M. G. Speranza. 2006. Worst-case analysis for split delivery vehicle routing problems. Transportation Science **40**, 226-234.

[19] Archetti, C., M. G. Speranza, M. W. P. Savelsbergh. 2008 An optimization-based heuristic for the split delivery vehicle routing problem. Transportation Science **42**, 22-31.

[20] Caplice, C. and Sheffi, Y. 2005. Combinatorial auctions of truckload transportation, in P. Cramton, Y. Shoham and R. Steinberg (eds), Combinatorial Auctions, MIT Press, Cambridge, MA, pp. 539-572.

[21] Ledyard, J. O., Olson, M., Porter, D., Swanson, J. A. and Torma, D. P. 2002, The first use of a combined-value auction for transportation services, Interfaces **32**(5), 4-12.

[22] Elmaghraby, W. and Keskinocak, P.: 2003, Combinatorial auctions in procurement, in C. Billington, T. Harrison, H. Lee and J. Neale (eds), The Practice of Supply Chain Management, Kluwer Academic Publishers, pp. 245-258.

[23] Song, J. and Regan, A. C. 2003. An auction based collaborative carrier network. working paper (UCI-ITS-LI-WP-03-6), UC Irvine, Institute of Transportation Studies.

[24] Song, J. and Regan, A. C. 2004. Combinatorial auctions for transportation service procurement - the carrier perspective, Transportation Research Record **1833**, 40-46.

[25] Song, J. and Regan, A. C. 2005. Approximation algorithms for the bid construction problem in combinatorial auctions for the procurement of freight transportation contracts, Transportation Research Part B: Methodological **39**(10), 914-933.

[26] Mullaseril PA, Dror M, Leung J. 1997. Split-delivery routing heuristics in livestock feed distribution. Journal of the Operational Research Society **48**(2), 107-116.

[27] Frizzell, P. W., J. W. Giffin. 1995. The split delivery vehicle scheduling problem with time windows and grid network distance. Comput. Oper. Res. **22**, 655-667.

[28] Sierksma, G., G. A. Tijssen. 1998. Routing helicopters for crew exchanges on off-shore locations. Ann. Oper. Res. **76**, 261-286.

[29] A. Lim., B. Rodregues, Z. Xu. 2008. Transportation procurement with seasonally-varying shipper demand and volume guarantees. Operations Research. **58**, 758-771.

[30] G. Laporte. 2007. What you should know about the vehicle routing problem. Naval Research Logistics **54**, 881-819.

[31] O. Bräysy, M. Gendreau. 2005a. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. Transportation Science **39**, 104-118.

[32] O. Bräysy, M. Gendreau. 2005b. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. Transportation Science **39**, 119-139.

[33] H. Psarafits. 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science **14**, 130-154.

[34] H. Psarafits. 1983. An exact algorithm for the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science **17**, 351-361.

[35] T. R. Sexton, D. B. Lawrence. 1983. The multiple-vehicle subscriber dial-a-ride problem. Working paper MS/S83-009, College of Business and Management, University of Maryland.

[36] T. R. Sexton, D. B. Lawrence. 1985a. Optimizing single vehicle many-to-many dial-a-ride problem with desired delivery time: I Scheduling. Transportation Science. **19**, 378-410.

[37] T. R. Sexton, D. B. Lawrence. 1985b. Optimizing single vehicle many-to-many dial-a-ride problem with desired delivery time: II Routing. Transportation Science. **19**, 411-435.

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

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

[40] Landrieu, A., Y. Mati, Z. Binder. 2001. A tabu search heuristic for the single vehicle pickup and delivery problem with time windows. J. Intelligent Manufacturing **12**, 497-508.

[41] Y. Dumas, J. Desrosiers, F. Soumis. A dynamic programming solution of the large scale single vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Science **16**, 301-325.

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

[43] Xu, H., Z.-L. Chen, S. Rajagopal, S. Arunapuram. 2003. Solving a practical pickup and delivery problem. Transportation Science **37**, 347C364.

[44] Lim, H., A. Lim, B. Rodrigues. 2002. Solving the pick up and delivery problem with time windows using "squeaky wheel" optimization with local search. Technical Report 2002, Paper 7-2002, Singapore Management University, Singapore.

[45] S. Ropke, D. Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **44**, 455-472.

Appendix A

Original Test Results

In this Appendix, we present the original results for these numerical tests in Chapter 4 and Chapter 5. Section A.1 presents the original results on different maximum allowed numbers of split loads, Section A.2 reports the original results on the smaller load size ranges, and Section A.3 reports the original results on the wider load size ranges.

A.1 For Different Maximum Allowed Numbers of Split Loads

In this section, original results for all the numerical tests on the 75-, 100-, and 125-request problem sets are presented. Each problem set consists of two different configurations of pickup and delivery locations, and for each location configuration, five different load configurations are generated and all the load sizes are within the range [0.51, 0.6]. Table A.1 presents the original results for the 75-request problem sets, Table A.2 presents the original results for the 100-request problem sets, and Table A.3 presents the original results for the 125-request problem sets.

Table A.1: Original test results for 75-request problem sets with different maximum allowed numbers of splits

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0	1.7765	5536.45	0		
2	12.7996	3502.03	30	2034.42	0.367459293
3	16.9956	3688.86	44	1847.59	0.333713842
4	19.0725	3654.03	43	1882.42	0.340004877
5	19.8257	3660.05	56	1876.4	0.338917537
6	16.6255	3620.81	48	1915.64	0.346005112
0	1.86588	5536.45	0		
2	7.03537	3956.82	24	1579.63	0.285314597
3	15.5283	3610.26	51	1926.19	0.347910665
4	21.0385	3671.02	43	1865.43	0.336936123
5	7.80619	3735.12	26	1801.33	0.325358307
6	19.1726	3796.28	43	1740.17	0.314311517
0	1.76685	5536.45	0		
2	8.30252	3922.7	28	1613.75	0.291477391
3	19.4366	3702.4	50	1834.05	0.331268231
4	6.83147	3893.73	23	1642.72	0.296709986
5	13.7884	3664.03	41	1872.42	0.338198665
6	29.6688	3605.32	53	1931.13	0.348802933
0	1.87707	5536.45	0		
2	15.152	3795.14	41	1741.31	0.314517425
3	21.6318	3656.84	41	1879.61	0.339497331
4	42.3977	3540.49	52	1995.96	0.360512603

Continued on next page

Table A.1 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
5	5.66081	3903.4	27	1633.05	0.294963379
6	22.095	3765.45	56	1771	0.319880068
0	1.84769	5536.45	0		
2	21.3867	3792.36	46	1744.09	0.315019552
3	17.4567	3689.09	39	1847.36	0.333672299
4	25.7982	3893.62	57	1642.83	0.296729854
5	39.4205	3539.06	52	1997.39	0.360770891
6	22.5993	3583.1	41	1953.35	0.352816335
0	1.7765	5536.45	0		
2	12.7996	3502.03	30	2034.42	0.367459293
3	16.9956	3688.86	44	1847.59	0.333713842
4	19.0725	3654.03	43	1882.42	0.340004877
5	19.8257	3660.05	56	1876.4	0.338917537
6	16.6255	3620.81	48	1915.64	0.346005112
0	1.86588	5536.45	0		
2	7.03537	3956.82	24	1579.63	0.285314597
3	15.5283	3610.26	51	1926.19	0.347910665
4	21.0385	3671.02	43	1865.43	0.336936123
5	7.80619	3735.12	26	1801.33	0.325358307
6	19.1726	3796.28	43	1740.17	0.314311517
0	1.76685	5536.45	0		
2	8.30252	3922.7	28	1613.75	0.291477391
3	19.4366	3702.4	50	1834.05	0.331268231

Continued on next page

Table A.1 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
4	6.83147	3893.73	23	1642.72	0.296709986
5	13.7884	3664.03	41	1872.42	0.338198665
6	29.6688	3605.32	53	1931.13	0.348802933
0	1.87707	5536.45	0		
2	15.152	3795.14	41	1741.31	0.314517425
3	21.6318	3656.84	41	1879.61	0.339497331
4	42.3977	3540.49	52	1995.96	0.360512603
5	5.66081	3903.4	27	1633.05	0.294963379
6	22.095	3765.45	56	1771	0.319880068
0	1.84769	5536.45	0		
2	21.3867	3792.36	46	1744.09	0.315019552
3	17.4567	3689.09	39	1847.36	0.333672299
4	25.7982	3893.62	57	1642.83	0.296729854
5	39.4205	3539.06	52	1997.39	0.360770891
6	22.5993	3583.1	41	1953.35	0.352816335

Table A.2: Original test results for 100-request problem sets with different maximum allowed numbers of splits

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0	1.7765	5536.45	0		
2	12.7996	3502.03	30	2034.42	0.367459293
3	16.9956	3688.86	44	1847.59	0.333713842

Continued on next page

Table A.2 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
4	19.0725	3654.03	43	1882.42	0.340004877
5	19.8257	3660.05	56	1876.4	0.338917537
6	16.6255	3620.81	48	1915.64	0.346005112
0	1.86588	5536.45	0		
2	7.03537	3956.82	24	1579.63	0.285314597
3	15.5283	3610.26	51	1926.19	0.347910665
4	21.0385	3671.02	43	1865.43	0.336936123
5	7.80619	3735.12	26	1801.33	0.325358307
6	19.1726	3796.28	43	1740.17	0.314311517
0	1.76685	5536.45	0		
2	8.30252	3922.7	28	1613.75	0.291477391
3	19.4366	3702.4	50	1834.05	0.331268231
4	6.83147	3893.73	23	1642.72	0.296709986
5	13.7884	3664.03	41	1872.42	0.338198665
6	29.6688	3605.32	53	1931.13	0.348802933
0	1.87707	5536.45	0		
2	15.152	3795.14	41	1741.31	0.314517425
3	21.6318	3656.84	41	1879.61	0.339497331
4	42.3977	3540.49	52	1995.96	0.360512603
5	5.66081	3903.4	27	1633.05	0.294963379
6	22.095	3765.45	56	1771	0.319880068
0	1.84769	5536.45	0		
2	21.3867	3792.36	46	1744.09	0.315019552
3	17.4567	3689.09	39	1847.36	0.333672299

Continued on next page

Table A.2 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
4	25.7982	3893.62	57	1642.83	0.296729854
5	39.4205	3539.06	52	1997.39	0.360770891
6	22.5993	3583.1	41	1953.35	0.352816335
0	2.53397	8821.61	0		
2	31.5689	5933.83	52	2887.78	0.327352944
3	13.5988	6035.12	31	2786.49	0.315870912
4	15.4478	5994.64	37	2826.97	0.320459644
5	77.3609	5692.01	69	3129.6	0.354765173
6	51.4204	5575.2	58	3246.41	0.36800652
0	2.49856	8821.61	0		
2	45.0785	5642.18	66	3179.43	0.360413802
3	6.82299	6757.88	17	2063.73	0.23394029
4	36.1735	6034.78	51	2786.83	0.315909454
5	37.2291	5851.78	53	2969.83	0.336653967
6	14.5761	6234.6	39	2587.01	0.29325826
0	2.76677	8821.61	0		
2	48.77	5602.29	61	3219.32	0.364935652
3	33.2558	5741.32	50	3080.29	0.349175491
4	12.2885	6252.26	35	2569.35	0.291256358
5	13.6478	6265.36	30	2556.25	0.289771368
6	8.65227	6955.64	14	1865.97	0.211522613
0	2.81267	8821.61	0		
2	22.6699	5934.64	44	2886.97	0.327261124

Continued on next page

Table A.2 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
3	25.8786	5904.87	53	2916.74	0.330635791
4	52.7174	5752.09	58	3069.52	0.347954625
5	18.4251	5852.56	51	2969.05	0.336565548
6	22.0169	5678.95	50	3142.66	0.356245629
0	2.50495	8821.61	0		
2	51.3834	5750.49	50	3071.12	0.348135998
3	39.0612	5822.55	40	2999.06	0.339967421
4	138.955	5665.66	72	3155.95	0.357752156
5	20.7408	5874.57	42	2947.04	0.334070538
6	20.998	5906.02	38	2915.59	0.330505429

Table A.3: Original test results for 125-request problem sets with different maximum allowed numbers of splits

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0	4.8763	11548.5	0		
2	36.5793	7713.83	44	3834.67	0.332049184
3	29.1905	8463.79	26	3084.71	0.267109148
4	76.3901	7401.97	77	4146.53	0.359053557
5	54.259	7567.5	71	3981	0.344720094
6	45.2287	7708.12	45	3840.38	0.33254362
0	5.81823	11548.5	0		
2	63.5377	7128.08	60	4420.42	0.382770057

Continued on next page

Table A.3 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
3	52.0173	7652.32	50	3896.18	0.337375417
4	36.9435	7559.63	49	3988.87	0.345401567
5	140.353	7157.14	79	4391.36	0.380253713
6	43.6222	7563.59	51	3984.91	0.345058666
0	7.6352	11548.5	0		
2	49.1231	7272.73	47	4275.77	0.370244621
3	93.6556	7010.76	69	4537.74	0.392928952
4	35.8491	7760.64	43	3787.86	0.327995844
5	125.307	7071.37	81	4477.13	0.387680651
6	220.463	6968.56	97	4579.94	0.396583106
0	6.59719	11548.5	0		
2	39.9363	7943.13	35	3605.37	0.312193791
3	58.3043	7289.62	55	4258.88	0.368782093
4	65.3268	7202.4	70	4346.1	0.376334589
5	32.215	7961.89	40	3586.61	0.310569338
6	50.1217	7225.64	67	4322.86	0.374322206
0	5.29178	11548.5	0		
2	52.3802	7325.78	63	4222.72	0.36565095
3	40.5436	8568.49	34	2980.01	0.258043036
4	83.8596	7139.21	73	4409.29	0.381806295
5	180.144	7193.03	90	4355.47	0.37714595
6	133.683	7279.14	78	4269.36	0.36968957
0	4.83736	10658.7	0		

Continued on next page

Table A.3 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
2	74.6972	7148.42	71	3510.28	0.329334722
3	82.9283	7412.6	76	3246.1	0.304549335
4	51.5441	7309.46	62	3349.24	0.314225937
5	81.3539	7450.89	75	3207.81	0.300956965
6	133.769	7489.39	105	3169.31	0.297344892
0	4.47688	10658.7	0		
2	135.756	7379.88	71	3278.82	0.307619128
3	132.648	7293.23	79	3365.47	0.315748637
4	196.911	7407.95	104	3250.75	0.304985599
5	137.023	7278.99	78	3379.71	0.317084635
6	85.3547	7489.83	69	3168.87	0.297303611
0	12.2387	10658.7	0		
2	62.8174	7395.25	63	3263.45	0.306177114
3	71.4281	7449.13	68	3209.57	0.301122088
4	83.5735	7465.78	68	3192.92	0.299559984
5	92.8698	7493.14	66	3165.56	0.296993067
6	168.496	7260.8	87	3397.9	0.318791222
0	9.6049	10658.7	0		
2	103.901	7193.29	76	3465.41	0.325125015
3	149.578	7161.18	89	3497.52	0.328137578
4	110.65	7178.61	65	3480.09	0.326502294
5	68.4235	7376.92	72	3281.78	0.307896835
6	93.3958	7303.86	77	3354.84	0.31475133
0	5.89349	10658.7	0		

Continued on next page

Table A.3 – continued from previous page

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
2	121.792	7283.43	89	3375.27	0.316668074
3	79.9549	7358.17	61	3300.53	0.309655962
4	99.0114	7268.1	59	3390.6	0.318106336
5	202.19	7323.11	81	3335.59	0.312945294
6	239.14	7258.31	82	3400.39	0.319024834

A.2 For Smaller Load Size Ranges

In this section, original results for all the numerical tests on the 75-, 100-, and 125-request problem sets are presented. Each problem set consists of two different configurations of pickup and delivery locations, and for each location configuration, five different load configurations are generated and all the load sizes are within the ranges from [0.11, 0.2] to [0.81, 0.9]. The results are presented in such a way that each result for the algorithm without split loads is presented before its corresponding algorithm with split loads. Table A.4 presents the original results for the 75-request problem sets, Table A.5 presents the original results for the 100-request problem sets, and Table A.6 presents the original results for the 125-request problem sets.

Table A.4: Original test results for 75-request problem sets on smaller load size ranges

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.11-0.20	4.15209	2133.91	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.55206	2133.91	0	0	0
	4.14034	2133.91	0		
	4.48524	2133.91	0	0	0
	5.3545	2133.91	0		
	4.24918	2133.91	0	0	0
	4.31103	2133.91	0		
	5.29806	2133.91	0	0	0
	3.67916	2133.91	0		
	4.00955	2133.91	0	0	0
0.11-0.20	3.15281	2228.21	0		
	3.17955	2228.21	0	0	0
	3.05949	2229.79	0		
	3.08345	2229.79	0	0	0
	3.53348	2229.79	0		
	3.82863	2229.79	0	0	0
	2.9471	2228.21	0		
	3.51346	2228.21	0	0	0
	3.19015	2228.21	0		
	3.72507	2228.21	0	0	0
0.11-0.20	5.48399	2414.09	0		
	3.79366	2414.09	0	0	0
	3.6421	2414.09	0		
	3.11561	2414.09	0	0	0
	2.9534	2414.09	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.3195	2414.09	0	0	0
	4.89867	2414.09	0		
	5.16378	2414.09	0	0	0
	3.37267	2415.94	0		
	3.69402	2415.94	0	0	0
0.21-0.30	3.89658	2073.35	0		
	7.3656	2080.47	6	-7.12	-0.003434056
	2.81083	2135.36	0		
	4.88904	2115.78	3	19.58	0.009169414
	2.69536	2197.11	0		
	3.1403	2197.11	0	0	0
	2.62077	2187.88	0		
	4.73762	2190.77	4	-2.89	-0.001320913
	4.05607	2094.19	0		
	8.30292	2088.24	6	5.95	0.002841194
0.21-0.30	2.93492	2234.61	0		
	5.07057	2234.61	0	0	0
	3.73356	2201.35	0		
	4.80556	2201.67	1	-0.32	-0.000145365
	6.43577	2151.73	0		
	6.7409	2128.17	3	23.56	0.010949329
	2.9734	2195.71	0		
	8.93416	2180.6	1	15.11	0.006881601
	3.81237	2154.49	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.771	2161.04	0	-6.55	-0.003040163
0.21-0.30	3.80447	2150.66	0		
	4.39929	2150.66	0	0	0
	3.80713	2287.16	0		
	5.81653	2287.16	0	0	0
	4.01085	2350.13	0		
	5.21375	2358.51	2	-8.38	-0.00356576
	2.92921	2292.45	0		
	2.91691	2292.45	0	0	0
	3.48458	2320.18	0		
	5.30374	2321.4	2	-1.22	-0.000525821
0.31-0.40	3.07184	3321.9	0		
	12.11	3176.67	19	145.23	0.043718956
	1.81257	3273.95	0		
	7.24184	3128	20	145.95	0.044579178
	2.20679	3265.97	0		
	9.85713	3028.42	22	237.55	0.0727349
	2.95545	3125.17	0		
	4.52449	3096.03	6	29.14	0.009324293
	2.25811	2650.46	0		
	3.89784	2603.97	4	46.49	0.017540351
0.31-0.40	2.066	2759.16	0		
	5.20949	2711.46	11	47.7	0.01728787
	2.45024	2753.04	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	8.44833	2661.75	11	91.29	0.033159707
	2.18601	2765.8	0		
	5.49288	2728.16	18	37.64	0.013609082
	2.6233	2766.73	0		
	3.58556	2704.62	9	62.11	0.022448884
	2.58534	2791.53	0		
	6.05404	2649.99	15	141.54	0.050703378
0.31-0.40	2.27785	2872.32	0		
	11.437	2803.81	14	68.51	0.023851799
	2.86791	2902.24	0		
	5.63453	2784.87	6	117.37	0.040441176
	2.5821	3002.47	0		
	4.32981	2860.71	11	141.76	0.04721446
	2.34372	2993.41	0		
	4.64498	2845.07	12	148.34	0.049555524
	1.85359	3014.74	0		
	7.381	2904.71	18	110.03	0.036497343
0.41-0.50	3.29863	3186.85	0		
	6.29048	3187.71	5	-0.86	-0.000269859
	4.66281	3186.85	0		
	4.054	3188.08	3	-1.23	-0.000385961
	4.13118	3186.85	0		
	6.61791	3190.32	6	-3.47	-0.001088849
	4.79476	3186.85	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	3.61495	3195.59	2	-8.74	-0.00274252
	2.43193	3186.85	0		
	5.0405	3189.24	2	-2.39	-0.000749957
0.41-0.50	3.11406	2783.3	0		
	5.45304	2785.76	4	-2.46	-0.000883843
	3.04954	2783.3	0		
	7.89817	2761.33	11	21.97	0.007893508
	2.60092	2783.3	0		
	3.20214	2783.94	2	-0.64	-0.000229943
	2.15793	2783.3	0		
	3.73545	2783.3	3	0	0
	5.25722	2783.3	0		
	2.15122	2783.3	0	0	0
0.41-0.50	3.32807	2899.19	0		
	5.63806	2901.81	1	-2.62	-0.000903701
	3.90724	2899.19	0		
	4.76785	2906.11	2	-6.92	-0.002386874
	3.3336	2899.19	0		
	5.4774	2903.64	3	-4.45	-0.001534911
	3.25058	2899.19	0		
	5.40938	2881.16	8	18.03	0.006218978
	3.72115	2899.19	0		
	4.66284	2884.36	2	14.83	0.005115222
0.51-0.60	2.17933	5536.45	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	25.5074	3636.4	43	1900.05	0.343189228
	2.06045	5536.45	0		
	22.9893	3692.46	54	1843.99	0.333063606
	1.50089	5536.45	0		
	26.3648	3561.98	53	1974.47	0.356631054
	2.07851	5536.45	0		
	24.8857	3598.81	50	1937.64	0.349978777
	2.12907	5536.45	0		
	15.9015	3549.04	45	1987.41	0.358968292
0.51-0.60	2.11356	5921.35	0		
	24.8001	3901.59	47	2019.76	0.341097892
	2.21583	5921.35	0		
	22.8894	4050.87	46	1870.48	0.315887424
	1.64355	5921.35	0		
	19.1874	3831.18	49	2090.17	0.352988761
	2.33106	5921.35	0		
	13.7139	4105.76	37	1815.59	0.306617579
	1.87999	5921.35	0		
	19.556	3991.45	47	1929.9	0.325922298
0.51-0.60	2.81699	7428.85	0		
	24.481	4698.51	49	2730.34	0.367531987
	1.53665	7428.85	0		
	27.8866	4708.95	46	2719.9	0.366126655
	1.96422	7428.85	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	21.3382	4506.01	40	2922.84	0.393444477
	2.09788	7428.85	0		
	13.1528	4587.85	39	2841	0.382427967
	1.4807	7428.85	0		
	32.3357	4382.77	76	3046.08	0.410033854
0.61-0.70	6.33792	6271.03	0		
	27.9643	4586.67	54	1684.36	0.268593835
	1.27459	6271.03	0		
	15.7352	4771.45	52	1499.58	0.239128181
	1.77857	6271.03	0		
	14.872	4853.52	37	1417.51	0.226041017
	1.77754	6271.03	0		
	20.87	4892.23	55	1378.8	0.219868188
	1.82821	6271.03	0		
	27.1986	4620.56	50	1650.47	0.26318962
0.61-0.70	1.85151	6272.79	0		
	18.7144	4477.45	43	1795.34	0.286210761
	2.03199	6272.79	0		
	25.1776	4447	43	1825.79	0.29106506
	1.78333	6272.79	0		
	13.8186	4346	34	1926.79	0.307166349
	1.94591	6272.79	0		
	11.314	4474.64	33	1798.15	0.286658728
	1.84326	6272.79	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	31.5149	4437.09	66	1835.7	0.2926449
0.61-0.70	2.03699	5949.5	0		
	17.1119	4601.87	27	1347.63	0.226511472
	1.60622	5949.5	0		
	6.56571	4670.48	23	1279.02	0.21497941
	1.8069	5949.5	0		
	11.1899	4610.25	36	1339.25	0.22510295
	1.32058	5949.5	0		
	18.2968	4492.31	55	1457.19	0.244926464
	1.60322	5949.5	0		
	10.2018	4564.87	36	1384.63	0.232730482
0.71-0.80	1.83836	6324.07	0		
	20.6817	5570.41	58	753.66	0.119173254
	1.82592	6324.07	0		
	51.4876	5516.46	77	807.61	0.127704153
	1.78696	6324.07	0		
	24.9245	5420.59	47	903.48	0.142863694
	1.54203	6324.07	0		
	11.2503	5636.22	34	687.85	0.108766981
	1.30655	6324.07	0		
	20.4239	5466.35	50	857.72	0.135627847
0.71-0.80	1.75172	6556.54	0		
	44.7682	5274.22	61	1282.32	0.195578766
	1.77984	6556.54	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	38.3099	5456.9	74	1099.64	0.167716509
	1.97899	6556.54	0		
	42.6104	5426.67	63	1129.87	0.172327173
	1.18891	6556.54	0		
	34.2878	5406.09	70	1150.45	0.175466023
	1.43087	6556.54	0		
	21.2008	5276.51	55	1280.03	0.195229496
0.71-0.80	1.39229	5723.74	0		
	49.9912	4933.31	83	790.43	0.138096769
	1.75389	5723.74	0		
	21.9168	4789.88	56	933.86	0.163155559
	2.19739	5723.74	0		
	36.9322	4661.62	61	1062.12	0.185563984
	1.28479	5723.74	0		
	41.0971	4951.56	76	772.18	0.134908294
	1.40266	5723.74	0		
	24.9523	4878.66	57	845.08	0.147644722
0.81-0.90	1.804	7068.23	0		
	18.5189	6614.13	38	454.1	0.064245221
	1.51443	7068.23	0		
	34.7811	6413.49	74	654.74	0.092631394
	1.46133	7068.23	0		
	16.3019	6535.34	29	532.89	0.075392283
	1.78584	7068.23	0		

Continued on next page

Table A.4 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	21.5387	6627.12	37	441.11	0.06240742
	1.777	7068.23	0		
	18.9118	6801.13	18	267.1	0.03778881
0.81-0.90	1.86941	5840.65	0		
	4.90311	5775.17	5	65.48	0.011211081
	1.77536	5840.65	0		
	23.9797	5549.64	41	291.01	0.049824934
	1.77138	5840.65	0		
	51.2465	5338.67	74	501.98	0.085945914
	1.42795	5840.65	0		
	7.643	5635.96	18	204.69	0.035045757
	1.60417	5840.65	0		
	17.8935	5532.62	52	308.03	0.052738993
0.81-0.90	1.90832	6845.15	0		
	18.2368	6153.41	46	691.74	0.101055492
	1.78872	6845.15	0		
	9.05178	6322.93	15	522.22	0.076290512
	1.51402	6845.15	0		
	13.4191	6247.8	18	597.35	0.087266167
	1.7825	6845.15	0		
	23.232	6360.26	61	484.89	0.070837016
	1.475	6845.15	0		
	17.0161	6039.88	40	805.27	0.117640957

Table A.5: Original test results for 100-request problem sets on smaller load size ranges

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.11-0.20	7.91706	2823.53	0		
	9.65498	2823.53	0	0	0
	9.60782	3013.96	0		
	12.3887	2936.62	6	77.34	0.025660593
	7.85844	2920.42	0		
	10.1757	2921.47	3	-1.05	-0.000359537
	8.84764	2874.34	0		
	8.23119	2874.34	0	0	0
	8.19824	2823.53	0		
	9.54575	2823.53	0	0	0
0.11-0.20	7.45925	2709.4	0		
	7.81542	2709.4	0	0	0
	7.71517	2687.35	0		
	11.626	2635.56	2	51.79	0.019271773
	7.67984	2687.35	0		
	8.09835	2687.35	0	0	0
	7.48232	2729.41	0		
	7.93534	2729.41	0	0	0
	8.08102	2687.35	0		
	9.38615	2687.35	0	0	0
0.11-0.20	7.46708	2976.51	0		
	7.07918	2976.51	0	0	0

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	7.05706	2976.51	0		
	7.37709	2976.51	0	0	0
	7.35949	2837.93	0		
	7.73571	2837.93	0	0	0
	7.12643	2957.49	0		
	7.41148	2957.49	0	0	0
	7.01206	2980.47	0		
	7.40459	2980.47	0	0	0
0.21-0.30	6.36053	3240	0		
	8.38402	3240	2	0	0
	7.31884	3261.79	0		
	6.10888	3261.79	0	0	0
	6.06748	3278.09	0		
	6.2995	3278.09	0	0	0
	6.22527	3437.24	0		
	6.40888	3437.24	0	0	0
	6.02133	3365.96	0		
	6.47486	3365.96	0	0	0
0.21-0.30	7.01513	3139.26	0		
	8.38592	3139.26	0	0	0
	6.49789	3119.51	0		
	8.50009	3119.51	0	0	0
	6.92011	3098.51	0		
	12.1882	3081.87	6	16.64	0.005370323

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	7.48605	3158.61	0		
	6.89625	3158.61	0	0	0
	6.42072	3148.36	0		
	10.8187	3143.27	5	5.09	0.001616715
	6.56822	3424.04	0		
	15.6523	3269	11	155.04	0.045279845
	6.96224	3531.99	0		
	5.48544	3531.99	0	0	0
	7.551	3318.3	0		
	13.9833	3154.89	9	163.41	0.049245095
	6.30139	3404.13	0		
	18.9968	3415.05	14	-10.92	-0.003207868
	6.19979	3395.96	0		
	6.15522	3395.96	0	0	0
0.31-0.40	5.20532	4569.48	0		
	26.5544	4156.66	33	412.82	0.090342884
	6.16519	4581.07	0		
	13.0861	4344.87	27	236.2	0.051560007
	6.03928	4331.06	0		
	13.2299	4206.71	14	124.35	0.028711216
	4.63767	4606.72	0		
	18.1986	4256	25	350.72	0.076132259
	5.49755	4451.09	0		
	30.4797	4075.96	32	375.13	0.084278233

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.31-0.40	5.74747	4954.02	0		
	21.0526	4643.69	29	310.33	0.062642056
	4.66205	4732.31	0		
	29.6002	4504.03	25	228.28	0.048238598
	4.72724	4768.67	0		
	21.8038	4538.52	29	230.15	0.048262933
	4.69074	4783.21	0		
	16.2781	4454.7	25	328.51	0.06867982
	4.71867	4943.7	0		
	13.5779	4691.48	20	252.22	0.051018468
0.31-0.40	5.44314	3749.7	0		
	18.548	3523.01	27	226.69	0.060455503
	5.36519	3833.9	0		
	28.7352	3656.12	18	177.78	0.046370537
	4.66191	3804.83	0		
	16.8499	3535.5	17	269.33	0.070786343
	5.28622	3818.26	0		
	15.4509	3636.21	15	182.05	0.047678786
	5.42989	3852.84	0		
	20.4073	3659.8	21	193.04	0.0501033
0.41-0.50	4.25788	4232.8	0		
	8.93895	4174.99	10	57.81	0.013657626
	4.82532	4232.8	0		
	5.44601	4232.8	0	0	0

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.27833	4232.8	0		
	9.14315	4144.96	3	87.84	0.020752221
	4.30264	4232.8	0		
	5.35756	4232.8	0	0	0
	4.66742	4232.8	0		
	8.60842	4191.35	3	41.45	0.009792572
0.41-0.50	4.77271	4246.01	0		
	5.62759	4246.01	0	0	0
	4.24639	4246.01	0		
	7.80065	4235.85	3	10.16	0.002392835
	5.89386	4246.01	0		
	4.7703	4246.01	0	0	0
	4.23586	4246.01	0		
	5.85848	4244.6	2	1.41	0.000332076
	4.32889	4246.01	0		
	5.20196	4246.01	0	0	0
0.41-0.50	4.43501	4147.65	0		
	6.20033	4147.65	0	0	0
	4.30738	4147.65	0		
	4.93616	4147.65	0	0	0
	4.33689	4147.65	0		
	8.65357	4140.77	4	6.88	0.001658771
	4.34865	4147.65	0		
	5.49731	4147.65	0	0	0

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.36219	4147.65	0		
	5.51674	4147.65	0	0	0
0.51-0.60	2.81691	7802.82	0		
	53.8795	5094.95	62	2707.87	0.347037353
	4.45853	7802.82	0		
	44.1533	5281.59	59	2521.23	0.323117796
	3.41686	7802.82	0		
	38.9525	5447.71	59	2355.11	0.301828057
	2.81339	7802.82	0		
	37.5629	5136.29	57	2666.53	0.341739269
	2.47636	7802.82	0		
	63.2693	5369.09	64	2433.73	0.311903901
0.51-0.60	2.74111	9209.16	0		
	34.4611	6326.94	64	2882.22	0.31297317
	2.47483	9209.16	0		
	25.9144	6155.6	51	3053.56	0.331578559
	2.8392	9209.16	0		
	44.9063	5989.89	55	3219.27	0.349572599
	3.37952	9209.16	0		
	77.1017	6088.71	71	3120.45	0.338841979
	2.42763	9209.16	0		
	51.3087	6142.21	81	3066.95	0.333032546
0.51-0.60	3.17896	8821.61	0		
	81.8202	5778.98	65	3042.63	0.344906429

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	3.16167	8821.61	0		
	43.2338	5891.49	68	2930.12	0.332152521
	3.13341	8821.61	0		
	44.4612	5790.28	52	3031.33	0.343625483
	3.15731	8821.61	0		
	46.2894	5614.37	57	3207.24	0.363566288
	3.15057	8821.61	0		
	40.8827	5584.46	59	3237.15	0.366956825
0.61-0.70	2.81846	7568.68	0		
	38.5689	5511.81	63	2056.87	0.27176073
	2.68082	7568.68	0		
	29.67	5535.1	53	2033.58	0.268683575
	2.80299	7568.68	0		
	58.2312	5553.58	87	2015.1	0.266241934
	2.65952	7568.68	0		
	29.6513	5586.33	55	1982.35	0.261914891
	2.44573	7568.68	0		
	24.9088	5480.3	51	2088.38	0.275923939
0.61-0.70	2.86331	8039.39	0		
	24.6619	5879.41	57	2159.98	0.268674613
	2.35295	8039.39	0		
	29.036	6016.37	57	2023.02	0.251638495
	2.46276	8039.39	0		
	39.8341	6004.41	57	2034.98	0.25312617

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	3.08335	8039.39	0		
	37.0587	6088.72	64	1950.67	0.242639056
	2.35729	8039.39	0		
	40.022	5983.67	69	2055.72	0.255705968
0.61-0.70	2.215	8151.18	0		
	26.8231	6291.12	45	1860.06	0.228195182
	2.79903	8151.18	0		
	44.1451	6189.37	59	1961.81	0.240678037
	2.91342	8151.18	0		
	44.1421	6200.49	58	1950.69	0.239313817
	2.699	8151.18	0		
	36.8819	6264.5	45	1886.68	0.231460966
	2.79924	8151.18	0		
	35.2157	6187.33	55	1963.85	0.240928307
0.71-0.80	2.69442	9100.05	0		
	63.168	7483.6	90	1616.45	0.177630892
	2.73487	9100.05	0		
	43.0897	7886.13	66	1213.92	0.133397069
	2.86044	9100.05	0		
	118.418	7424.52	100	1675.53	0.184123164
	2.70266	9100.05	0		
	94.7218	7691.42	95	1408.63	0.154793655
	2.22092	9100.05	0		
	42.7128	7567.35	68	1532.7	0.168427646

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.71-0.80	2.73072	6880.38	0		
	32.1867	6197.86	43	682.52	0.099198009
	2.8492	6880.38	0		
	12.1214	6157.35	24	723.03	0.105085766
	2.74468	6880.38	0		
	40.3629	6076.92	55	803.46	0.116775527
	2.81278	6880.38	0		
	56.5071	6158.99	79	721.39	0.104847407
	2.64398	6880.38	0		
	48.765	6119.83	74	760.55	0.110538953
0.71-0.80	2.74508	7172.14	0		
	77.0018	6428.14	91	744	0.103734729
	3.03506	7172.14	0		
	46.3596	6410.57	69	761.57	0.106184486
	3.01527	7172.14	0		
	88.7651	6350.54	73	821.6	0.114554373
	4.73709	7172.14	0		
	42.9706	6331.44	63	840.7	0.117217455
	2.65849	7172.14	0		
	31.6899	6323.87	55	848.27	0.118272928
0.81-0.90	2.9376	9115.64	0		
	14.3027	8947.73	9	167.91	0.01841999
	2.59944	9115.64	0		
	3.43643	9115.67	0	-0.03	-3.29105E-06

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	2.30297	9115.64	0		
	2.92426	9115.72	0	-0.08	-8.77613E-06
	2.3669	9115.64	0		
	4.38061	9115.79	2	-0.15	-1.64552E-05
	3.10114	9115.64	0		
	6.2431	9115.93	1	-0.29	-3.18135E-05
0.81-0.90	3.13033	8922.72	0		
	7.89004	8922.85	3	-0.13	-1.45695E-05
	2.98319	8922.72	0		
	5.63574	8922.9	1	-0.18	-2.01732E-05
	2.86648	8922.72	0		
	4.1576	8923.53	0	-0.81	-9.07795E-05
	2.95876	8922.72	0		
	5.88956	8923.1	2	-0.38	-4.25879E-05
	3.10095	8922.72	0		
	3.53582	8924.35	0	-1.63	-0.00018268
0.81-0.90	3.01943	9023.99	0		
	6.04646	9014.28	3	9.71	0.001076021
	2.97091	9023.99	0		
	14.048	9014.45	4	9.54	0.001057182
	3.04131	9023.99	0		
	15.0786	8942.01	14	81.98	0.009084673
	2.90841	9023.99	0		
	5.79935	9024	1	-0.01	-1.10816E-06

Continued on next page

Table A.5 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	2.54512	9023.99	0		
	3.82855	9024.03	0	-0.04	-4.43263E-06

Table A.6: Original test results for 125-request problem sets on smaller load size ranges

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.11-0.20	11.5194	3658.76	0		
	11.6973	3658.76	0	0	0
	12.9578	3649.21	0		
	13.6388	3649.21	0	0	0
	13.5165	3474.17	0		
	14.3191	3474.17	0	0	0
	18.9712	3555.15	0		
	11.9128	3555.15	0	0	0
	15.3818	3474.68	0		
	14.1136	3474.68	0	0	0
0.11-0.20	14.1413	2988.48	0		
	14.4894	2988.48	0	0	0
	13.9378	3010.35	0		
	14.3469	3010.35	0	0	0
	14.7281	2973.7	0		
	14.7072	2973.7	0	0	0

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	14.4697	2931.94	0		
	14.58	2931.94	0	0	0
	14.0555	2911.77	0		
	14.7189	2911.77	0	0	0
0.11-0.20	12.999	2762.86	0		
	13.5534	2762.86	0	0	0
	13.1911	2820.48	0		
	13.3878	2820.48	0	0	0
	13.3771	2781.12	0		
	13.9639	2781.12	0	0	0
	13.3207	2773.14	0		
	13.8228	2773.14	0	0	0
	13.009	2825.83	0		
	13.6752	2825.83	0	0	0
0.21-0.30	11.8052	3731.9	0		
	19.9273	3664.2	3	67.7	0.018140893
	14.4446	3823.26	0		
	27.2806	3767.47	4	55.79	0.014592259
	10.853	3664.63	0		
	15.1138	3556.15	8	108.48	0.029601897
	16.5103	3829.17	0		
	41.3336	3597.96	24	231.21	0.060381231
	10.5504	3862.17	0		
	15.132	3759.35	3	102.82	0.026622339

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.21-0.30	10.2235	4333.07	0		
	37.9233	4110.84	27	222.23	0.051286963
	10.549	4229.56	0		
	28.1934	3963.65	11	265.91	0.062869424
	10.8706	4496.8	0		
	35.2476	4092.73	20	404.07	0.089857232
	12.0225	4157.74	0		
	12.4225	4063.56	4	94.18	0.022651729
	10.5047	4488.27	0		
	31.9145	4221.42	15	266.85	0.059454979
0.21-0.30	10.4955	4185.41	0		
	24.0372	3967.46	13	217.95	0.052073751
	11.2249	4015.23	0		
	27.0322	3922.49	13	92.74	0.023097058
	9.73235	3810.78	0		
	14.2711	3769.44	3	41.34	0.010848173
	10.0047	4021.75	0		
	28.7667	3877.92	13	143.83	0.035763038
	9.44704	4034.96	0		
	15.8471	3891.61	5	143.35	0.035526994
0.31-0.40	9.13985	5655.83	0		
	23.3983	5270.21	25	385.62	0.068180974
	8.25984	5787.31	0		
	115.764	5323.16	46	464.15	0.080201337

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	8.7096	5733.78	0		
	32.4674	5337.88	32	395.9	0.069046946
	11.7265	5644.48	0		
	42.0277	5268.97	41	375.51	0.066526943
	9.45313	5811.19	0		
	24.5241	5311.76	36	499.43	0.08594281
0.31-0.40	7.94457	5492.03	0		
	30.5067	5098.74	37	393.29	0.071611044
	11.0107	5232.91	0		
	19.4243	4913.97	22	318.94	0.060948879
	9.34574	5762.94	0		
	47.0932	5139.62	44	623.32	0.108160071
	9.01085	5596.07	0		
	37.5157	5187.12	24	408.95	0.073078071
	8.01732	5504.15	0		
	47.529	4978.01	43	526.14	0.095589691
0.31-0.40	8.84592	5634.65	0		
	40.3198	5145.04	28	489.61	0.086892709
	9.23927	5432.25	0		
	36.7584	5086.65	41	345.6	0.063620047
	9.15476	5553.48	0		
	61.2035	5088.22	33	465.26	0.083778099
	8.75961	5695.5	0		
	19.1105	5220.7	25	474.8	0.083364059

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	7.61246	5631.74	0		
	36.9547	5195.07	40	436.67	0.077537315
0.41-0.50	8.41458	5886.93	0		
	10.1997	5888.14	2	-1.21	-0.00020554
	9.86417	5886.93	0		
	10.1588	5886.93	2	0	0
	8.38073	5886.93	0		
	10.2295	5887.08	2	-0.15	-2.54802E-05
	8.50815	5886.93	0		
	14.468	5888.15	4	-1.22	-0.000207239
	8.54737	5886.93	0		
	11.6599	5886.93	2	0	0
0.41-0.50	9.70316	5886.93	0		
	23.9222	5845.13	16	41.8	0.007100475
	9.11915	5886.93	0		
	30.4437	5857.52	13	29.41	0.004995813
	13.9945	5886.93	0		
	9.36049	5887.35	2	-0.42	-7.13445E-05
	10.4682	5886.93	0		
	11.021	5892.3	4	-5.37	-0.00091219
	8.76495	5886.93	0		
	13.5778	5887.42	2	-0.49	-8.32352E-05
0.41-0.50	12.5909	5886.93	0		
	15.3659	5893.4	2	-6.47	-0.001099045

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	15.2322	5886.93	0		
	15.1249	5851.5	12	35.43	0.006018417
	14.9093	5886.93	0		
	12.6903	5887.08	3	-0.15	-2.54802E-05
	8.52536	5886.93	0		
	13.9365	5886.93	5	0	0
	10.1344	5886.93	0		
	15.9705	5784.08	5	102.85	0.017470906
0.51-0.60	4.13398	11548.5	0		
	109.42	7124.39	79	4424.11	0.383089579
	4.98657	11548.5	0		
	164.607	6827.91	103	4720.59	0.408762177
	4.28788	11548.5	0		
	81.5478	6998.03	91	4550.47	0.394031259
	4.28645	11548.5	0		
	145.449	7009.09	89	4539.41	0.393073559
	4.09243	11548.5	0		
	115.532	6806.48	78	4742.02	0.410617829
0.51-0.60	4.20777	10658.7	0		
	147.649	7197.41	106	3461.29	0.324738477
	3.71087	10658.7	0		
	77.4357	7307.08	85	3351.62	0.314449229
	4.66583	10658.7	0		
	73.3299	7196.98	74	3461.72	0.324778819

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	6.53853	10658.7	0		
	96.7593	7267.31	86	3391.39	0.318180454
	7.2392	10658.7	0		
	89.7834	7162.02	69	3496.68	0.328058769
0.51-0.60	4.11005	9490	0		
	86.3537	6384.47	65	3105.53	0.32724236
	4.4027	9490	0		
	137.973	6266.02	67	3223.98	0.33972392
	4.24167	9490	0		
	129.569	6108.33	85	3381.67	0.356340358
	4.14164	9490	0		
	55.5845	6462.51	68	3027.49	0.319018967
	4.67401	9490	0		
	91.2725	6296.81	78	3193.19	0.336479452
0.61-0.70	4.40314	11246	0		
	46.293	8259.42	60	2986.58	0.265568202
	3.74975	11246	0		
	34.9571	8456	53	2790	0.248088209
	3.60125	11246	0		
	133.655	8342.54	96	2903.46	0.25817713
	3.66783	11246	0		
	58.3415	8404.39	70	2841.61	0.252677396
	3.66323	11246	0		
	40.268	8447.05	66	2798.95	0.248884048

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.61-0.70	4.32314	10282.3	0		
	54.2896	7436.92	62	2845.38	0.276726024
	4.58572	10282.3	0		
	60.3071	7390.82	60	2891.48	0.281209457
	4.31195	10282.3	0		
	32.3328	7568.61	42	2713.69	0.263918579
	4.46996	10282.3	0		
	57.4024	7496.03	63	2786.27	0.270977311
	4.4361	10282.3	0		
	83.4196	7482.95	88	2799.35	0.272249399
0.61-0.70	4.47853	10264.4	0		
	59.807	7689.83	76	2574.57	0.250825182
	4.85132	10264.4	0		
	74.1696	7758.34	69	2506.06	0.244150657
	5.2955	10264.4	0		
	76.1706	7866.61	82	2397.79	0.233602549
	4.9786	10264.4	0		
	37.8327	7790.56	52	2473.84	0.241011652
	7.48034	10264.4	0		
	50.4148	7843.5	64	2420.9	0.23585402
0.71-0.80	4.33415	10463.1	0		
	127.995	8054.11	81	2408.99	0.230236737
	5.31898	10463.1	0		
	149.961	8522.75	92	1940.35	0.185446952

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	6.71647	10463.1	0		
	195.14	8314.42	130	2148.68	0.205357877
	4.43546	10463.1	0		
	144.176	8386.37	111	2076.73	0.19848133
	4.82622	10463.1	0		
	99.1799	8347.43	80	2115.67	0.20220298
0.71-0.80	4.28131	9871.36	0		
	97.3193	8468.95	96	1402.41	0.14206857
	4.64581	9871.36	0		
	122.539	8484.55	86	1386.81	0.140488241
	5.30979	9871.36	0		
	143.342	8440.83	112	1430.53	0.144917215
	4.9706	9871.36	0		
	91.2145	8536.45	89	1334.91	0.135230607
	5.1132	9871.36	0		
	147.623	8344.61	94	1526.75	0.154664605
0.71-0.80	4.63181	11518.3	0		
	143.244	9411.24	113	2107.06	0.182931509
	4.49809	11518.3	0		
	218.959	9513.3	125	2005	0.174070826
	4.63181	11518.3	0		
	141.148	9413.03	113	2105.27	0.182776104
	6.28745	11518.3	0		
	76.1562	9541.56	79	1976.74	0.171617339

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.49809	11518.3	0		
	218.959	9513.3	125	2005	0.174070826
0.81-0.90	5.22899	8315.09	0		
	44.5381	8257.55	24	57.54	0.006919949
	4.49756	8315.09	0		
	20.7954	8260.32	9	54.77	0.00658682
	4.52883	8315.09	0		
	15.985	8261.88	7	53.21	0.006399209
	4.53313	8315.09	0		
	18.5878	8174.04	21	141.05	0.016963136
	4.28281	8315.09	0		
	55.2116	8245.06	30	70.03	0.008422038
0.81-0.90	4.60223	9650.42	0		
	13.0948	9579.17	5	71.25	0.007383098
	4.66781	9650.42	0		
	8.69813	9618.42	1	32	0.003315918
	4.53251	9650.42	0		
	8.3024	9705.78	1	-55.36	-0.005736538
	4.76206	9650.42	0		
	28.4312	9513.35	16	137.07	0.014203527
	4.63863	9650.42	0		
	15.6478	9526.07	6	124.35	0.01288545
0.81-0.90	4.74953	9668.5	0		
	56.9582	8940.37	35	728.13	0.07530951

Continued on next page

Table A.6 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.54897	9668.5	0		
	29.8857	9394.12	18	274.38	0.028378756
	7.13987	9668.5	0		
	23.1868	9021.1	12	647.4	0.066959715
	4.62078	9668.5	0		
	42.6591	9194.53	25	473.97	0.049022082
	4.02151	9668.5	0		
	19.6304	9280.27	11	388.23	0.040154109

A.3 For Wider Load Size Ranges

In this section, original results for all the numerical tests on the 75-, 100-, and 125-request problem sets are presented. Each problem set consists of two different configurations of pickup and delivery locations, and for each location configuration, five different load configurations are generated and all the load sizes are within the four wider ranges, [0.1, 0.5], [0.5, 1.0], [0.3, 0.7], and [0.1, 1.0]. The results are presented in the same way as that is used for the numerical results on the smaller ranges. Table A.7 presents the original results for the 75-request problem sets, Table A.8 presents the original results for the 100-request problem sets, and Table A.9 presents the original results for the 125-request problem sets.

Table A.7: Original test results for 75-request problem sets on wider load size ranges

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.1-1.0	2.75865	4751.39	0		
	16.704	4537.05	24	214.34	0.04511101
	3.36853	4553.54	0		
	25.3221	4249.28	25	304.26	0.066818344
	2.69974	4742.87	0		
	12.7788	4549.58	11	193.29	0.040753805
	3.37557	4566.26	0		
	12.5548	4225.51	17	340.75	0.074623434
	3.05322	4804.71	0		
	6.17126	4664.39	19	140.32	0.029204676
0.1-1.0	2.83704	3525.66	0		
	15.3155	3275.41	22	250.25	0.070979618
	3.1355	3845.13	0		
	18.0298	3560.58	34	284.55	0.0740027
	2.91609	3680.53	0		
	11.4277	3506.46	18	174.07	0.047294819
	3.73953	3941.44	0		
	16.6182	3639.99	20	301.45	0.076482199
	2.95557	4094.03	0		
	28.4981	3813.56	27	280.47	0.06850707
0.1-1.0	3.32851	4009.17	0		
	14.5271	3776.36	27	232.81	0.058069376

Continued on next page

Table A.7 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	2.64635	3884.35	0		
	10.055	3670.14	7	214.21	0.055146936
	2.71854	3628.69	0		
	12.5323	3489.1	6	139.59	0.038468428
	2.34473	4068.13	0		
	12.0734	3930.98	23	137.15	0.033713279
	2.19114	3532.07	0		
	8.42546	3417.45	4	114.62	0.032451225
0.1-0.5	4.12528	2607.52	0		
	18.3602	2503.69	23	103.83	0.039819445
	3.65717	2330.39	0		
	9.30145	2244.54	14	85.85	0.036839327
	3.67384	2487.63	0		
	5.15046	2403.67	1	83.96	0.033751
	3.11183	2296.19	0		
	8.45993	2216.08	7	80.11	0.034888228
	3.3959	2434.26	0		
	11.98	2284.46	13	149.8	0.061538209
0.1-0.5	3.88923	2023.8	0		
	8.52402	1878.64	6	145.16	0.071726455
	3.63539	2087.5	0		
	9.11162	1998.83	9	88.67	0.042476647
	3.94578	2309.71	0		
	6.89754	2248.86	10	60.85	0.026345299

Continued on next page

Table A.7 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	3.62708	2205.9	0		
	13.3603	2155.18	13	50.72	0.022992883
	3.06561	2179.79	0		
	11.1812	2093.6	11	86.19	0.039540506
0.1-0.5	3.62604	2968.86	0		
	7.45393	2803.11	10	165.75	0.05582951
	3.19942	2708.72	0		
	21.1528	2628.72	16	80	0.029534245
	3.16347	2771.91	0		
	9.45313	2688.04	5	83.87	0.030257115
	3.85306	2721.03	0		
	5.77618	2667.52	4	53.51	0.019665347
	3.15505	2699.74	0		
	7.15152	2658.76	5	40.98	0.015179239
0.5-1.0	1.78099	5990.97	0		
	18.296	4867.55	58	1123.42	0.187518883
	1.78961	5990.97	0		
	18.3622	4987.6	46	1003.37	0.167480391
	1.75861	5990.97	0		
	27.766	4694.1	53	1296.87	0.216470789
	3.76663	5990.97	0		
	14.4754	5035.71	40	955.26	0.159449972
	1.27104	5990.97	0		
	23.9787	4750.88	43	1240.09	0.206993191

Continued on next page

Table A.7 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.5-1.0	1.81851	5669.89	0		
	9.28808	4411.46	35	1258.43	0.221949632
	1.48902	5669.89	0		
	19.6705	4549.47	42	1120.42	0.197608772
	1.21743	5669.89	0		
	21.6279	4395.4	51	1274.49	0.224782139
	1.20378	5669.89	0		
	15.6266	4556.28	44	1113.61	0.19640769
	1.19623	5669.89	0		
	17.4843	4470.88	55	1199.01	0.211469711
0.5-1.0	1.79156	6005	0		
	23.5494	5250.48	49	754.52	0.125648626
	1.70297	6005	0		
	15.0586	4819.84	42	1185.16	0.197362198
	1.71843	6005	0		
	21.8864	5153.08	34	851.92	0.141868443
	1.69714	6005	0		
	25.9092	5007.91	38	997.09	0.166043297
	1.73032	6005	0		
	10.7109	5002.93	31	1002.07	0.166872606
0.3-0.7	3.45476	4562.15	0		
	7.63138	4397.28	12	164.87	0.036138663
	3.19056	4847.74	0		
	7.93268	4635.16	9	212.58	0.043851362

Continued on next page

Table A.7 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	2.63746	5109.86	0		
	9.24333	4768.12	21	341.74	0.066878545
	3.34636	4894.01	0		
	10.9911	4713.3	13	180.71	0.03692473
	3.15354	5307.2	0		
	7.88207	4869.78	10	437.42	0.082420109
0.3-0.7	3.13687	3338.73	0		
	10.5178	3155.98	16	182.75	0.054736382
	2.57324	3365.91	0		
	9.47829	3221.9	9	144.01	0.042784864
	2.61575	3714.36	0		
	18.5792	3391.95	22	322.41	0.086800956
	3.21634	3474.05	0		
	6.16903	3297.9	8	176.15	0.050704509
	2.54868	3459.06	0		
	5.33081	3300.78	10	158.28	0.045758096
0.3-0.7	2.66829	4205.82	0		
	18.2292	3874.66	27	331.16	0.07873851
	2.57158	4003.93	0		
	12.7177	3830.88	18	173.05	0.043220036
	3.28888	3994.74	0		
	29.8936	3578.33	31	416.41	0.104239575
	2.27273	4099.86	0		
	18.298	3677.18	31	422.68	0.103096203

Continued on next page

Table A.7 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	2.51742	3913.34	0		
	10.8199	3680.72	12	232.62	0.059442829

Table A.8: Original test results for 100-request problem sets on wider load size ranges

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.1-1.0	5.78299	5537.34	0		
	13.6476	5351.4	19	185.94	0.0335793
	4.56195	5297.57	0		
	19.8833	5075.17	19	222.4	0.041981512
	5.54443	5434.27	0		
	17.3692	5268.67	16	165.6	0.030473274
	5.41659	5055.69	0		
	19.9833	4846.08	20	209.61	0.041460216
	4.70454	4872.96	0		
	19.0373	4611.34	24	261.62	0.053688107
0.1-1.0	7.04941	4399.93	0		
	18.4946	4234.88	26	165.05	0.03751196
	4.63024	4148.92	0		
	28.5872	3907.65	19	241.27	0.058152483
	6.50215	4098.89	0		
	51.5415	3812.53	36	286.36	0.069862817

Continued on next page

Table A.8 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	5.21569	4078.5	0		
	17.645	3919.95	29	158.55	0.038874586
	3.94731	4163.83	0		
	15.1336	3977.28	14	186.55	0.044802502
0.1-1.0	4.34498	5463.23	0		
	48.3169	5137.59	40	325.64	0.059605764
	5.73414	5163.3	0		
	19.4235	4943.18	22	220.12	0.04263165
	5.51699	5100.3	0		
	23.0248	4772.15	30	328.15	0.064339353
	4.28441	5215.2	0		
	31.837	4942.6	34	272.6	0.052270287
	5.40177	4944.16	0		
	30.7674	4757.86	33	186.3	0.037680819
0.1-0.5	7.4327	4177.54	0		
	18.093	4066.95	17	110.59	0.026472517
	6.14238	4103.84	0		
	15.944	3832.41	14	271.43	0.066140493
	6.25431	4152.18	0		
	19.3563	4044.75	18	107.43	0.025873156
	8.13818	4027.39	0		
	17.2017	3858.17	15	169.22	0.042017287
	6.10473	4344.33	0		
	19.434	4227.29	18	117.04	0.026940863

Continued on next page

Table A.8 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.1-0.5	6.29321	3782.33	0		
	16.107	3691.22	10	91.11	0.024088327
	5.70091	3808.15	0		
	17.2477	3733.44	17	74.71	0.01961845
	5.71788	3923.97	0		
	18.8472	3831.43	15	92.54	0.023583259
	5.51993	3639.71	0		
	13.3181	3550.39	7	89.32	0.024540417
	5.58723	3523.72	0		
	16.1119	3423.45	14	100.27	0.028455723
0.1-0.5	5.566	3850.18	0		
	14.0752	3744.75	10	105.43	0.027383135
	6.74644	3676.19	0		
	13.4217	3593.81	10	82.38	0.02240907
	5.4873	3619.4	0		
	14.6123	3494.73	15	124.67	0.034444936
	5.78755	3669.26	0		
	12.1456	3486.25	6	183.01	0.049876542
	5.6357	3665.15	0		
	11.817	3586.78	12	78.37	0.021382481
0.5-1.0	3.13883	9821.55	0		
	26.1056	7605.49	52	2216.06	0.22563241
	3.0381	9821.55	0		
	37.1122	7717.18	52	2104.37	0.214260478

Continued on next page

Table A.8 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.04695	9821.55	0		
	91.2051	7575.89	78	2245.66	0.228646191
	2.84844	9821.55	0		
	63.0915	7484.42	63	2337.13	0.237959385
	3.96498	9821.55	0		
	43.9444	7440.87	67	2380.68	0.242393512
0.5-1.0	2.85217	9791.94	0		
	28.5427	7853.19	49	1938.75	0.197994473
	2.39632	9791.94	0		
	45.3911	7411.06	65	2380.88	0.243146915
	2.75811	9791.94	0		
	42.273	7618.66	68	2173.28	0.221945804
	2.47108	9791.94	0		
	16.0452	7874.68	47	1917.26	0.195799811
	2.61554	9791.94	0		
	45.9507	7429.11	66	2362.83	0.241303562
0.5-1.0	3.72153	7709.54	0		
	120.377	6252.83	90	1456.71	0.188949016
	2.88466	7709.54	0		
	31.5058	6245.32	56	1464.22	0.189923134
	3.0891	7709.54	0		
	64.5364	6219.31	76	1490.23	0.193296876
	4.47849	7709.54	0		
	28.5762	6242.26	46	1467.28	0.190320045

Continued on next page

Table A.8 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	2.22226	7709.54	0		
	35.5826	6199.53	48	1510.01	0.195862529
0.3-0.7	5.02758	4141.27	0		
	13.2748	3937.39	18	203.88	0.049231274
	4.8383	4240.55	0		
	13.6746	3934.68	15	305.87	0.072129794
	7.78203	4158.28	0		
	26.361	3887.51	22	270.77	0.065115865
	4.25255	4152.63	0		
	36.911	3900.91	38	251.72	0.060617007
	6.02539	3971.94	0		
	12.496	3811.95	11	159.99	0.040280065
0.3-0.7	7.4366	5578.62	0		
	31.9365	5132.63	40	445.99	0.079946295
	6.95518	5313.61	0		
	17.625	5027.64	27	285.97	0.053818402
	5.32398	5503.5	0		
	17.1034	5200.16	25	303.34	0.055117652
	6.89982	5426.17	0		
	18.5622	5026.38	28	399.79	0.073678119
	6.61519	4958.6	0		
	24.2826	4789.42	26	169.18	0.034118501
0.3-0.7	5.61286	6395.99	0		
	34.6965	5755.39	40	640.6	0.100156504

Continued on next page

Table A.8 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	5.88763	5856.24	0		
	29.8269	5379.84	28	476.4	0.081349125
	6.28159	6135.63	0		
	44.3583	5505.31	36	630.32	0.102731097
	9.31084	6097.16	0		
	58.2476	5538.61	52	558.55	0.091608224
	7.35799	6044.45	0		
	20.8535	5556.4	20	488.05	0.080743492

Table A.9: Original test results for 125-request problem sets on wider load size ranges

Max Splits Allowed	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.1-1.0	9.51468	7321.32	0		
	68.0916	6944.37	33	376.95	0.051486617
	10.2223	6678.78	0		
	33.91	6397.35	22	281.43	0.042137935
	11.747	6730.25	0		
	37.9265	6555.52	26	174.73	0.025961888
	11.3672	6870.68	0		
	79.6477	6580.91	38	289.77	0.042174865
	12.9814	7108.04	0		
	20.8363	6875.58	14	232.46	0.032703811

Continued on next page

Table A.9 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
0.1-1.0	9.42585	7662.46	0		
	43.5878	7300.98	28	361.48	0.04717545
	10.7186	7413.73	0		
	14.3509	6998.27	12	415.46	0.056039268
	9.45163	7336.87	0		
	28.5348	7040.56	28	296.31	0.040386432
	10.2695	7118.99	0		
	50.3736	6669.13	31	449.86	0.063191548
	9.75323	6668.12	0		
	30.2647	6322.62	13	345.5	0.051813705
0.1-1.0	10.4039	6610.73	0		
	30.3418	6397.77	22	212.96	0.032214294
	8.48573	7072.84	0		
	32.9545	6873.09	24	199.75	0.028241838
	8.48573	7072.84	0		
	32.9545	6873.09	24	199.75	0.028241838
	9.27977	7165.33	0		
	41.7361	6885.07	36	280.26	0.039113342
	8.11325	7276.77	0		
	54.5706	6953.33	43	323.44	0.044448292
0.1-0.5	13.8052	3753.35	0		
	20.9997	3659.49	9	93.86	0.025006994
	13.0689	4036.71	0		
	32.7937	3919.27	16	117.44	0.029092999

Continued on next page

Table A.9 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	13.1047	3753.09	0		
	25.8798	3603.6	20	149.49	0.039831179
	13.0157	3825.75	0		
	29.2107	3674.67	27	151.08	0.039490296
	10.5982	3932.9	0		
	25.6809	3831.08	20	101.82	0.025889293
0.1-0.5	12.2386	3740.91	0		
	27.7522	3563.18	13	177.73	0.04750983
	9.44012	3725.01	0		
	17.0327	3664.37	5	60.64	0.016279151
	10.5081	3787.3	0		
	15.2203	3686.95	5	100.35	0.026496449
	10.4666	3819.55	0		
	23.8767	3700.33	13	119.22	0.031213101
	11.9842	3731.36	0		
	12.6629	3647.53	3	83.83	0.022466339
0.1-0.5	10.596	3730.12	0		
	15.0881	3561.34	4	168.78	0.045247874
	10.6654	3665.02	0		
	13.9178	3588.07	3	76.95	0.020995793
	10.3152	3928.67	0		
	15.053	3850.48	10	78.19	0.01990241
	11.9851	3690.39	0		
	26.6269	3541.82	13	148.57	0.040258618

Continued on next page

Table A.9 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	10.8122	3723.94	0		
	20.0504	3599.16	9	124.78	0.033507522
0.5-1.0	3.73637	9217.02	0		
	65.5488	7567.42	70	1649.6	0.178973247
	4.67497	9217.02	0		
	94.1849	7623.11	100	1593.91	0.172931164
	4.54023	9217.02	0		
	109.23	7615.48	70	1601.54	0.173758981
	5.57636	9217.02	0		
	52.7201	7428.69	81	1788.33	0.19402475
	4.94004	9217.02	0		
	72.0719	7641.99	65	1575.03	0.17088278
0.5-1.0	3.63596	8657.13	0		
	152.751	6985.13	97	1672	0.1931356
	4.09211	8657.13	0		
	92.2896	7017.42	89	1639.71	0.189405727
	3.72622	8657.13	0		
	102.235	6912.27	102	1744.86	0.201551784
	3.60229	8657.13	0		
	82.024	6919.05	86	1738.08	0.200768615
	3.73926	8657.13	0		
	42.9393	6780.31	69	1876.82	0.216794711
0.5-1.0	4.10829	8879.17	0		
	68.6495	7273.94	72	1605.23	0.180786042

Continued on next page

Table A.9 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	4.06951	8879.17	0		
	40.8004	7262.02	59	1617.15	0.18212851
	3.65268	8879.17	0		
	84.7895	7351.4	70	1527.77	0.172062254
	3.8466	8879.17	0		
	91.4477	7277.59	96	1601.58	0.180374967
	4.35992	8879.17	0		
	85.3072	7243.19	74	1635.98	0.184249203
0.3-0.7	8.65174	6817.94	0		
	20.935	6364.36	31	453.58	0.066527426
	7.54979	6564.06	0		
	27.4304	6167.17	20	396.89	0.060464103
	8.10458	6589.06	0		
	27.9438	5987.98	23	601.08	0.091223938
	9.47348	6718.09	0		
	20.046	6297.4	18	420.69	0.062620477
	8.53615	6814.26	0		
	25.9914	6433.79	27	380.47	0.055834383
0.3-0.7	8.89768	6723.87	0		
	24.4032	6066.01	38	657.86	0.097839488
	11.384	6576.9	0		
	36.2563	6066.5	28	510.4	0.077604951
	8.85976	6568.97	0		
	27.863	6136.29	24	432.68	0.065867252

Continued on next page

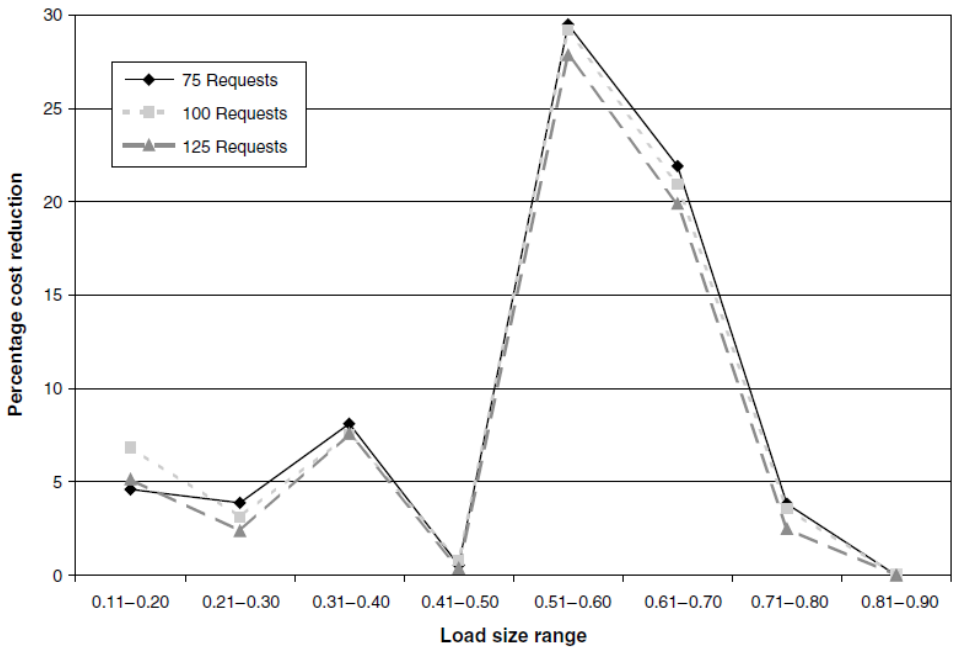
Table A.9 – continued from previous page

Load Range	CPU Time	Cost	Splits	Cost Reduction	Percentage
	8.46466	6591.99	0		
	35.3784	6077.45	29	514.54	0.078055337
	10.7166	6475.39	0		
	16.4842	5973.36	17	502.03	0.077528921
0.3-0.7	7.14452	6822.04	0		
	28.1075	6272.07	35	549.97	0.080616648
	10.2081	6435.15	0		
	17.5244	5999.64	15	435.51	0.067676744
	11.0794	6933.05	0		
	25.5554	6460.16	31	472.89	0.068208076
	7.01046	6757.38	0		
	80.7439	6291.55	39	465.83	0.068936481
	8.35574	6859.01	0		
	14.7995	6382.15	21	476.86	0.069523153

Appendix B

Key Results in Nowak. et. al. 2008

In this Appendix, we attach the key results in Nowak. et. al. 2008 [6] (NOWAK hereafter) for a reference and comparison with our results as reported in the main body of this dissertation and in Appendix A.



Average Percentage Cost Reduction with Split Loads for Each Load Range Tested for the 75-, 100-, and 125-Request Problem Sets

Figure B.1: Numerical Test Results on Smaller Load Size Ranges in NOWAK

Table B.1: Average CPU Time, No. of Splits with Split Loads on Smaller Load Size Ranges in NOWAK

	Problem Size					
	75		100		125	
Load Range	Time	Splits	Time	Splits	Time	Splits
0.11-0.20	10.2	0.3	24.8	0.7	67.2	1.3
0.21-0.30	10.5	1.3	20.8	2	53.8	2.1
0.31-0.40	15.1	8.1	28.2	10.6	56.3	14.5
0.41-0.50	6.6	0	13.6	0.4	23.6	1.3
0.51-0.60	25.5	28.4	56.2	39.3	95.9	47.1
0.61-0.70	18.3	21.7	38.8	28.9	68.7	36.7
0.71-0.80	5.8	7	10.9	7.3	16.6	7.7
0.81-0.90	4.9	0	8.8	0	13.7	0

Table B.2: Average CPU Time, Cost Reduction with Split Loads on Wider Load Size Ranges in NOWAK

	Problem Size					
	75		100		125	
Load Range	Time	Cost Reduction	Time	Cost Reduction	Time	Cost Reduction
0.1-1.0	7.6	4.1	15.8	3.6	25.6	3.6
0.1-0.5	9.4	3.4	19.3	3.5	34.1	2.6
0.5-1.0	11.2	10.2	27.7	9.9	48.7	10.8
0.3-0.7	11.1	6.1	24.9	5.8	41.8	5.3