The Pennsylvania State University

The Graduate School

College of Engineering

SOLVING THE SPLIT DELIVERY VEHICLE ROUTING PROBLEM

A Dissertation in

Industrial Engineering and Operations Research

by

Joseph Hubert Wilck, IV

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2009

The dissertation of Joseph Hubert Wilck, IV was reviewed and approved* by the following:

Tom M. Cavalier
Professor of Industrial Engineering
Dissertation Advisor, Chair of Committee

M. Jeya Chandra
Professor of Industrial Engineering
Professor in Charge of Academic Programs and Graduate Program Coordinator

Tao Yao
Assistant Professor of Industrial Engineering

Marc E. McDill
Associate Professor of Forest Resource Management

Paul M. Griffin
Peter and Angela Dal Pezzo Department Head Chair

*Signatures are on file in the Graduate School

ABSTRACT

The Split Delivery Vehicle Routing Problem (SDVRP) allows customers to be assigned to multiple routes. A flow formulation and a set-covering based formulation are presented for the SDVRP. A construction heuristic procedure is developed and used to build an initial population for a hybrid genetic algorithm. The construction heuristic, along with the set-covering based formulation, are used to develop a column generation procedure. These three procedures for the SDVRP are compared and computational results are given for data sets from previous literature. With respect to the total travel distance and computation time, the three procedures compare favorably versus previously published methods. The column generation procedure provides lower bounds (dual bounds), thereby reducing the Duality GAP to less than 5% within 5101.699 seconds of computer time for 32 test problems with up to 288 customers and 216 vehicle routes. These results represent a significant improvement when compared to previously published methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (CONTINUED)

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my committee members. Dr. Chandra, you personally recruited me to Penn State, gave me Teaching Assistant and Lecturer opportunities, and offered exuberant pride in maintaining a challenging academic program. I appreciate your effort, dedication, and heart for students. Dr. Yao, I am grateful for you serving on my committee and working with me during my first weeks at Penn State. Dr. McDill, you introduced me to an entire field of research and study in Forestry and Harvest Scheduling, and I hope to contribute to that field in the future. And finally, Dr. Cavalier, my advisor, I thank you for challenging me in teaching, coursework, and research; and for your patience that I tested from time-to-time. I will always remember our conversations that swayed from sports to operations research to politics to academic life, etc. I wish I could have learned more from you as a teacher, advisor, and colleague. In conclusion, I am proud to have each of you as my committee members, and I appreciate your efforts, suggestions, and time spent guiding me through this dissertation and research.

Second, I would like to acknowledge people who have helped me along-the-way. Dr. Meller and Dr. Ellis, my M.S. advisors at Virginia Tech; Dr. Fraticelli, one of my professors at Virginia Tech, who first suggested that I apply to Penn State (her B.S. and M.S. *alma mater*); Dr. Koubek, the Department Head during my formative years at Penn State; the administrative staff, Ms. Covasa, Ms. Frazier, Ms. Fuoss, Ms. Hosterman, Ms. Williams, and many others who looked out for my fellow students and I; Ms. Reed in the College of Engineering Dean's Office; and the faculty members at Virginia Tech and at Penn State who guided me through courses, projects, and research to facilitate my learning and professional activities. I would like to acknowledge the Material Handling Institute of America (MHIA) for scholarship support and the Free Software Foundation for GNU FORTRAN.

Third, I would like to thank my friends and colleagues from the various organizations I worked with and various places I have either lived or worked. The College of Engineering Faculty Council, Industrial Engineering Graduate Association, Engineering Graduate Student Council, New Life, Baptist Student Union (Virginia Tech), Merck, Philip Morris, Volvo, and my fellow classmates, officemates, roommates, and colleagues from both Virginia Tech and Penn State. I would also like to express my gratitude to my teachers and friends from Farmville, VA. Since I have many friends, I will only acknowledge one by name, Brian Roy Bluhm (July 19, 1981 - April 16, 2007), the biggest Detroit Tigers fan I have ever known, an intelligent and thoughtful friend, and gracious teammate. Collectively, to my friends and colleagues, it is a joy to know all of you and I appreciate the memories of Bible studies, conversations, intramural games, working out, and fun we have endured together.

Fourth, I am blessed with a wonderful family. My grandparents laid a firm foundation built on love and hard work. Joe, Walter, and Ethel have passed, but I will continue to extend their memory and legacy; and I will continue to learn from Catherine. My parents always believed in me and challenged me to pursue my dreams. My mom, Susan, my biggest fan and greatest supporter; has such a passion for people. My stepdad, Bob, keeps me grounded; especially when I tend to theorize a little too much. My dad, Joe, shares my passion for knowledge and sports. My stepmom, Eileen, has kindly accepted me into her family. My siblings provide inspiration to me in various ways. My younger sister, Elizabeth, whom I am very proud, probably knows me better than anyone, and challenges me with her intelligence, love, and selflessness. My younger brothers, Jonathon and Justin, you both provide me with motivation to make the world a better place, and I encourage you to get the most out of your youth. My extended family, I am grateful for your love and support. My wife's parents, Newell and Nita, wife's sister, Jennifer, and my wife's extended family and friends; thank you for accepting me into your family and I appreciate your commitment to my wife and I. And finally, to my wife, Karen, I love you, I realize I am a difficult person to live with and love; I value your patience and devotion. Overall, I love my family and I will strive to express that love whenever possible.

Finally, I acknowledge Jesus Christ, my personal Lord and Savior. Without Him I would be living aimlessly and with little hope. Thank You!

DEDICATION

To my family: past, present, and future - with Love.

Chapter 1

Introduction

The Vehicle Routing Problem (VRP) is a prominent combinatorial optimization problem in distribution planning. The VRP extends the Traveling Salesman Problem (TSP) to multiple routes where a subset of customers is serviced on each route. Various forms of the VRP are studied for both theoretical and practical purposes. The solution quality of these problems is essential to the logistics industry when considering transportation costs. Dantzig and Ramser (1959) first exhibited these cost savings by providing a formulation and solution approach for gasoline distribution. Toth and Vigo (2002a) estimate that computer-generated solutions for distribution problems reduce transportation costs by 5%-20%, and this is a significant decrease since transportation costs represent between 10%-20% of the final cost of a unit.

A well-studied form of the VRP is the Capacitated Vehicle Routing Problem (CVRP). The CVRP establishes routes to a set of customers from a single depot to minimize the total distance traveled. The problem is constrained by the vehicle capacity while meeting customer demand. By definition, a customer is on exactly one route. Review literature for the VRP includes a survey by Toth and Vigo (2002b) for exact solution procedures, whereas Cordeau et al. (2002, 2005) survey heuristic procedures, and a recent book edited by Golden et al. (2008). Earlier review literature includes a book edited by Golden and Assad (1988) and a bibliography by Laporte and Osman

(1995). Toth and Vigo (2002a) examine exact and heuristic procedures for the traditional VRP, the CVRP, and other variants. These variants include features such as pickup and delivery, time windows, and stochastic travel times. A variant that has received less attention in the VRP literature is the Split Delivery Vehicle Routing Problem.

The Split Delivery Vehicle Routing Problem (SDVRP) is a relaxation of the CVRP. One constraint set of the CVRP states that a customer is serviced by exactly one vehicle route. As shown in Figure **1-1**, the SDVRP allows customers to be serviced by more than one vehicle route, which usually reduces the number of routes and the total distance traveled (Dror and Trudeau 1989). Generally, the SDVRP takes more computer time to solve than the CVRP since the feasible space is larger (Dror and Trudeau 1989, Dror and Trudeau 1990, Dror et al. 1994). Dror and Trudeau (1990) show that the SDVRP is NP – hard.

This dissertation develops, for the SDVRP, a heuristic algorithm to construct feasible solutions, a genetic algorithm, a set-covering formulation, and a column generation procedure. Computational experience is provided by using data sets from previous literature. Due to the complexity of the SDVRP, obtaining small optimality gaps will not be possible for large problem instances. Toth and Vigo (2002a) state that CVRP instances with only fifty customers are taxing for exact solution methods, and the SDVRP is more complicated than the CVRP.

The remainder of this thesis is organized as follows. The next chapter reviews the literature for the SDVRP, genetic algorithms, and column generation. An SDVRP flow formulation, SDVRP construction heuristic, and computational experience are provided in Chapter 3. The SDVRP genetic algorithm and computational experience are given in

Chapter 4. A set-covering formulation, column generation procedure, and computational experience are provided in Chapter 5. A summary is offered in Chapter 6.

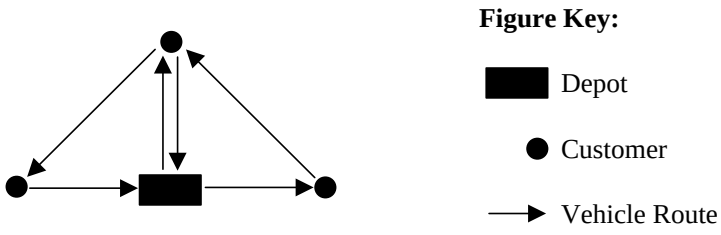

Figure **1-1**: SDVRP solution for three customers and two vehicle routes.

Chapter 2

Literature Review

The primary topic of this thesis is a VRP variant, the SDVRP. Standard forms of the VRP have been studied for decades as shown in the seminal paper by Dantzig and Ramser (1959) and an algorithm developed by Clarke and Wright (1964). Recent surveys by Toth and Vigo (2002b) and Cordeau et al. (2002, 2005) examine exact and heuristic procedures of the VRP, respectively. Golden and Assad (1988) and Toth and Vigo (2002a) edited books solely devoted to the VRP and its variants, and Laporte and Osman (1995) provide an extensive bibliography. The remainder of the chapter is organized as follows. Section 2.1 focuses on the literature pertaining to the SDVRP, Section 2.2 discusses the use of genetic algorithms for variants of the VRP, and column generation is explained in Section 2.3. A summary is given in Section 2.4.

2.1 Split Delivery Vehicle Routing Problem

The SDVRP is appropriate for many CVRP applications where customers can be visited more than once. Numerous applications for the CVRP are noted in the literature (Toth and Vigo 2002b, Golden and Assad 1988), with the primary emphasis being the distribution of various goods. Dror and Trudeau (1989, 1990) formally introduced the SDVRP. The primary motivation to split a customer's demand over multiple routes is to reduce the travel distance and the number of vehicle routes. If each vehicle has the same

capacity, then the minimum number of routes is the total demand divided by the vehicle capacity rounded up to the nearest integer. Archetti and Speranza (2008) and Gulczynski et al. (2008) recently surveyed the SDVRP literature.

Dror and Trudeau (1989, 1990) proved, given that the distance between nodes follows the triangle inequality, there exists an optimal solution where no two routes can have more than one split demand point in common, and that there exists an optimal solution with no k-split cycles (for any k). The triangle inequality is defined as:

Given nodes i, j, and k. Let the travel distance from i to j be denoted by c_{ij}. The triangle inequality states that: $c_{ij} \leq c_{ik} + c_{kj}$ for all i, j, and k.

A k-split cycle, as developed by Dror and Trudeau (1990), is defined as:

> Given k demand points $p_1, p_2, \ldots, p_k$ and k vehicle routes. If route 1 includes the points p_1 and p_2, route 2 includes the points p_2 and p_3, …, route $k-1$ includes the points p_{k-1} and p_k, and route k includes the points p_k and p_1; then the subset of points $p_1, p_2, \ldots, p_k$ are considered a k-split cycle. Therefore, the points $p_1, p_2, \ldots, p_k$ all receive split deliveries by the k vehicle routes.

Based on these proofs, Dror and Trudeau (1990) devised a heuristic to solve the SDVRP given an initial CVRP solution by splitting the demand of a node to fill the routes to capacity. This algorithm allows for additional routes over the minimum number of routes. The heuristic does not eliminate k-split cycles; however, no 2-split or 3-split cycles were observed in any of the resulting solutions.

Dror et al. (1994) extended the formulation of Dror and Trudeau (1989, 1990) with additional constraints, and developed a constraint relaxation branch and bound

algorithm. The directed arc formulation allows the number of vehicle routes to vary from the minimum number to the number of customers. The additional constraints strengthen the subtour elimination constraints, eliminate some fractional cycles when the binary decision variables are relaxed, and impose an upper bound on the number of arcs used in an optimal solution. Results of the algorithm were given for problems with 10, 15, and 20 customers and optimality gaps were reported. The optimality gap was within 10% for 10 customer problems, 20% for 15 customer problems, and 30% for 20 customer problems. However, in order to reduce the number of constraints produced, arcs over a certain length were eliminated from consideration.

The results from Dror and Trudeau (1989, 1990) and Dror et al. (1994) show that the percent reduction in travel distance, when compared to the CVRP, is most prominent among problems where customers have high demands (i.e., more than 10% of the vehicle capacity). The results also indicate that good solutions to the SDVRP tend to have routes that sweep geographic areas, and that customers with split demand over multiple routes usually have above average demand. The results showed that the SDVRP solution used fewer routes than the CVRP solution; however, the SDVRP solution did not always use the minimum number of routes. The computer time to obtain the SDVRP solution was longer than the time to compute the CVRP solution.

Frizzell and Giffin (1992, 1995) solved the SDVRP with time windows on grid networks and developed heuristics to generate solutions. Archetti et al. (2006a) developed a tabu search algorithm for the SDVRP. Archetti et al. (2006b) analyzed the worst-case properties of the SDVRP, where they show that the savings are at most 50% over the optimal CVRP solution. Archetti et al. (2008) discussed when demand splitting

is most beneficial. Their results indicated that demand splitting is best when the mean customer demand is just over half of the vehicle capacity and when customer demand variance is low. Lee et al. (2006) developed a dynamic programming model for the vehicle routing problem with split pick-ups with an uncountable (infinite) number of state and action spaces.

Belenguer et al. (2000) performed a polyhedral study on the SDVRP to produce lower bounds through formulating the problem with undirected arcs by assuming symmetric distances (i.e., $c_{ij} = c_{ji}$). Using a cutting-plane algorithm in conjunction with a relaxed formulation, they were able to obtain feasibility gaps within 12% for problems with 50, 75, and 100 customers. They did not report computation time, only the number of iterations and the number of cuts.

Jin et al. (2007a) developed a two-stage algorithm to solve the SDVRP using valid inequalities. Jin et al. (2007b, 2008) presented a column generation procedure that provides comparable lower and upper bounds for the data sets developed by Belenguer et al. (2000). Jin et al. (2007b, 2008) allowed for additional routes above the minimum number of vehicle routes. Chen et al. (2007) presented a mixed integer program and a variable length record-to-record travel algorithm. They also provided computational results for existing and new data sets. Archetti et al. (2006), Chen et al. (2007), and Chen (2007) used the Clarke – Wright Algorithm (Clarke and Wright, 1964) for the CVRP to provide an initial starting solution for the SDVRP. Burrows (1988) showed that the SDVRP could be modified by splitting customer demand into smaller quantities on the same node, and then solved using CVRP methods. Recent theses addressing the SDVRP, or close variants, include Liu (2005), Nowak (2005), and Chen (2007).

Direct applications of the SDVRP have been noted in the literature. Mullaseril et al. (1997) modeled a cattle feed distribution problem as an SDVRP with time windows. They developed a solution algorithm that uses *k*-split interchanges and route addition. Sierksma and Tijssen (1998) model a helicopter crew-scheduling problem using an SDVRP model, and they developed two solution approaches. The first used a relaxed linear program and column generation scheme to find a solution. The second used a cluster-and-route procedure. Song et al. (2002) modeled a newspaper distribution process as a SDVRP and solved using a two-phase procedure. The first phase allocates customers using a binary program and the second phase generates vehicle routes.

2.2 Genetic Algorithm procedures for Vehicle Routing Problems

A genetic algorithm is a global search procedure that solves problems by emulating evolution. A pure genetic algorithm uses reproduction and mutation to develop a new generation of solutions from the current generation of solutions. The constraints of the VRP do not allow the application of pure genetic algorithms without an additional step to ensure feasibility.

The book by Goldberg (1989) describes the solution process of genetic algorithms. The Fundamental Theorem of Genetic Algorithms states the conditions in order to achieve a global optimal solution. These conditions describe the breeding process and insist that better solutions (or patterns) remain in future generations while weaker solutions (or patterns) are eliminated from future generations. The typical genetic algorithm follows these basic steps (Winston and Venkataramanan, 2003):

Step 1: Generate an initial population of solutions.

Step 2: Evaluate each solution using a fitness function (or objective function).

Step 3: Select, using probability and randomness, solutions as parents for the new generation. The best solutions (in terms of fitness or objective) have a higher probability of being selected than poor solutions.

Step 4: Use the parent solutions from Step 3 to produce the next generation (called offspring). Often this procedure is referred to as crossover or recombination.

Step 5: Randomly alter the new generation by mutation.

Step 6: Repeat Steps 2 through 5 until a stopping criteria is met.

The procedure outlined above is a basic genetic algorithm. Procedures for generating feasible offspring and feasibly mutating the new generation are problem-specific. For example, given a problem that can be represented by a binary string with four values. Then the following example can occur: Parent 1: 0-1-1-0 and Parent 2: 1-0-1-1.

A crossover will occur between any of the four values (i.e., between the first and second, between the second and third, and between the third and fourth). Based on probability (e.g., generating a random number between 0 and 1), the crossover point is chosen as between the second value and the third value. Thus the crossover will take 0-1 from Parent 1 and switch it with 1-0 from Parent 2, resulting in two offspring: Child 1: 1-0-1-0 and Child 2: 0-1-1-1.

The mutation stage will then, using probability, randomly select a small number of offspring (if any at all) and change a small portion of position values. For example, suppose that based on probability, Child 2 is to be mutated in the second position. Originally, Child 2 is 0-1-1-1; and with mutation she is 0-0-1-1.

The preceding example assumed a binary string problem structure with no limiting constraints. Unfortunately, applying genetic algorithms directly to VRP variants is difficult due to the constraints of the problems. Therefore, additional steps or changes are necessary to ensure feasibility of created solutions. For VRP variants the reproduction stage (i.e., Step 4) can be modified to ensure feasible solutions or an additional step can be added after the mutation stage (i.e., Step 5) to fix infeasible solutions.

Gendreau et al. (1997) described three approaches to the VRP with time windows, where each approach ensures feasibility differently. The first approach ensures feasibility while sacrificing the evolutionary aspects of the genetic algorithm; thus, it is considered a hybrid approach. This approach does not allow infeasible reproduction or mutation. The second approach applies a genetic algorithm to the partitioning of customers to routes, but the routes are solved separately to ensure feasibility. The third approach applies a genetic algorithm directly, but incorporates a post-processing step to ensure feasibility. The third approach yielded the best results in terms of the objective; however, it was computationally expensive. Gendreau et al. (2002) stated that accounting for the constraints of the VRP makes a genetic algorithm computationally expensive.

Alvarenga et al. (2007) developed a two-phase approach for the VRP with time windows by using a hybrid genetic algorithm and a set-partitioning method to generate

routes. The two-phase approach yielded good solutions when compared to best known solutions. Solution time was not compared or reported, but the genetic algorithm was given a time limit of 60 minutes.

Baker and Ayechew (2003) developed a pure genetic algorithm and a hybrid genetic algorithm for the CVRP while constraining the maximum distance of a route. The pure genetic algorithm produced poor solutions, when compared to previous simulated annealing and tabu search methods. The hybrid genetic algorithm provided comparable, although not superior, solutions when compared to previous methods in a reasonable amount of computer time. The hybrid genetic algorithm applied neighborhood search procedures to ensure a feasible solution.

Wang et al. (2008) developed a genetic algorithm for the generalized orienteering problem. The orienteering problem is a VRP variant where a start point and an end point are specified and other points have associated scores. The objective is to determine a path that maximizes the score while adhering to a time constraint. The generalized orienteering problem (GOP) adds a level of complexity where there are numerous attributes at a specific point that represent the total score. The GOP is similar to a VRP with one vehicle route. The genetic algorithm developed by Wang et al. (2008) compared favorably to an artificial neural network solution procedure. The genetic algorithm procedure initially allowed infeasible solutions, but then corrected the solutions by truncating them to accommodate the time constraint.

Jeon et al. (2007) consider the VRP with multiple depots and up to two deliveries per node (i.e., split delivery). They developed a pure genetic algorithm and a hybrid genetic algorithm. The hybrid genetic algorithm ensured that no infeasible solutions

were generated, and the hybrid genetic algorithm provided consistently better results in terms of objective value. Computation time was not provided for the pure genetic algorithm.

2.3 Column Generation procedures for Vehicle Routing Problems

Column generation is an optimization procedure that solves problems without enumerating all of the variable columns, extending the decomposition algorithm in linear programming. Column generation was developed by Gilmore and Gomory (1961) to solve the cutting-stock problem, and more recent applications are discussed in the book edited by Desaulniers et al. (2005). Vanderbeck and Wolsey (1996) describe a modified column generation approach that combines branching and subproblem modification. Barnhart et al. (1998) describe a branch-and-price column generation procedure that is useful for large integer programming problems, such as VRPs. Bradley et al. (1977) explain the column generation procedure for a linear program as follows. The original linear program is the *overall problem* and it is solved by iteratively solving the *restricted master problem* and the *subproblem*. The *restricted master problem* includes a portion of the variables from the *overall problem*. *Note: shadow prices, dual prices, and dual variables are considered synonyms for the purposes of this section.*

To begin, assume a set of variables in the *overall problem* are nonbasic, and set them equal to zero. The remaining variables are basic to the *overall problem*. Solve the *restricted master problem* with only the basic variables identified in the *overall problem*; generating the a_{ij} matrix only for the given basic variables. Once the optimal solution is

known for the *restricted master problem*, given the basic variables, the optimal shadow prices are then also know. The basic variables, nonbasic variables, and optimal shadow prices for the *restricted master problem* provide a primal feasible and complementary slackness satisfactory solution to the *overall problem*. Thus, checking for dual feasibility is required to determine if those variables and shadow prices are optimal to the *overall problem*. Checking for this dual feasibility is considered the *subproblem*. The *subproblem* identifies which variables need to be added to the *restricted master problem*. The *subproblem* is the primary difference between optimization problem types; whether they are linear, nonlinear, integer, etc.

In general, column generation can be performed without initial basic and nonbasic variables for the *restricted master problem*. Artificial variables can be used initially, or the *restricted master problem* can begin with no basic variables (i.e., not existing until the *subproblem* identifies a set of variables to add to the *restricted master problem*). For the SDVRP we will begin with an initial set of variables identified by previous heuristic methods (e.g., construction heuristic, genetic algorithm). Jin et al. (2007b, 2008) used a column generation procedure to solve the SDVRP and used the Clarke – Wright (1964) procedure to find initial variables to be basic and nonbasic for the *overall problem*.

Indexed Sets:

$i = 1, 2, \ldots, m$; constraint index

$j = 1, 2, \ldots, J$; variable index for the *restricted master problem*

$s = 1, 2, \ldots, n$; variable index for the optimal solution to the *subproblem*

Parameters:

m : number of constraints

n : number of total variables

b_i : right-hand side coefficient for the i^{th} constraint

a_{ij} : coefficient for constraint the i^{th} constraint and the j^{th} variable

c_j : cost coefficient for the j^{th} variable

J : current number of variables in the *restricted master problem* (variables are initially established by some method, for the SDVRP procedure discussed in later sections initial variables will be identified by a construction heuristic)

Variables:

π_i^J : optimal shadow price for the i^{th} constraint with J variables in the *restricted master problem*

x_j : decision variable for the j^{th} variable

Let variables $x_{J+1}, x_{J+2}, \ldots, x_n$ be nonbasic and zero.

Objectives:

z : optimal solution to the *overall problem*

z^J : optimal solution to the *restricted master problem* with J variables

v^J : optimal solution to the *subproblem* with J variables in the *restricted master problem*

Overall Problem:

Objective:

Minimize $z = \sum_{j=1}^{n} c_j x_j$

Constraints:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ \forall \ i = 1, 2, \ldots, m$$

$$x_j \geq 0 \ \forall \ j = 1, 2, \ldots, n$$

Column Generation Procedure:

Step 1: Solve the *restricted master problem*.

Restricted Master Problem:

Objective:

Minimize $z^J = \sum_{j=1}^{J} c_j x_j$

Constraints:

$$\sum_{j=1}^{J} a_{ij} x_j = b_i \ \forall \ i = 1, 2, \ldots, m$$

$$x_j \geq 0 \ \forall \ j = 1, 2, \ldots, J$$

Let $\pi_1^J, \pi_2^J, \ldots, \pi_m^J$ be the optimal shadow prices to the *restricted master problem*.

Step 2: Solve the *subproblem.*

Subproblem:

$$v^J = \underset{1 \le j \le n}{Minimize} \left\{ c_j - \sum_{i=1}^{m} \pi_i^J a_{ij} \right\}$$

Let $v^J = c_s - \sum_{i=1}^{m} \pi_i^J a_{is}$ be the optimal solution to the *subproblem*.

Step 3: If $v^J < 0$, then increase J to $J+1$ and add variable x_s to the *restricted master subproblem* and proceed to Step 1. If $v^J \ge 0$, then the solution to the *restricted master problem* is optimal to the *overall problem* with $x_{J+1} = x_{J+2} = \ldots = x_n = 0$ and $z = z^J$; STOP.

The efficiency of the column generation procedure is dependent upon two major factors, most notably (Bradley et al., 1977):

1. Obtaining an optimal solution without adding too many variables x_s to the solution. Otherwise, the *restricted master problem* becomes difficult to solve for similar reasons the *overall problem* is difficult to solve.
2. Solving the *subproblem* efficiently.

To combat the first factor, obtaining an optimal solution without adding too many variables x_s to the solution, Bramel and Simchi-Levi (2002) suggest adding multiple columns to the *restricted master problem* during each iteration. They examined solution

methods for the CVRP using column generation and a set-covering formulation. Their results indicate for the CVRP that multiple columns be identified by the *subproblem* in Step 2 and added to the *restricted master problem* in Step 3. Adding multiple columns in a single iteration generally decreased the computer runtime. Barnhart et al. (1998) propose steps to reduce the number of variables in the *restricted master problem* by eliminating nonbasic variables.

To combat the second factor, it is problem specific. For example, Jin et al. (2007b, 2008) for the SDVRP used a limited-search-with-bound algorithm to solve their *subproblem*. Bradley et al. (1977) and Winston and Venkataramanan (2003) for the cutting-stock problem reduce the *subproblem* to a knapsack problem. The knapsack problem can then be solved in a number of ways including branch-and-bound and dynamic programming. Bramel and Simchi-Levi (2002) for the CVRP solve the *subproblem* directly by calculating the values based on a predetermined set of nodes (some subset of the total node population); thus, they do not find a true optimal solution to the *overall problem*.

Column generation provides an optimal solution for linear programs and a dual bound for integer programs when integer constraints are relaxed. An integer feasible solution can be obtained by rounding, a branch-and-bound procedure, or using cutting planes.

Jin et al. (2007b, 2008) presented a column generation procedure for the SDVRP based on a flow representation on the problem. They allow more than the minimum number of vehicles to satisfy the demand, while focusing on minimizing the total travel distance. They solve their *subproblem* using a limited-search-with-bound algorithm.

This procedure provides both an upper and lower bound for the SDVRP; however, since more than the minimum number of vehicles are allowed these bounds are not directly comparable to other results since the feasible space is larger and the total travel distance could be smaller since more vehicles are allowed. Gendreau et al. (2006) provided a similar column generation approach for the SDVRP with time windows.

2.4 Literature Review Summary

This chapter provides background information on the SDVRP, including its origins, applications, and current status. Also discussed in this chapter are the basics for two solution procedures, genetic algorithms and column generation. Genetic algorithms have not been successfully applied to the SDVRP. Column generation has been applied to solve the SDVRP, but results of this dissertation show improvement over previous results by using a stronger formulation for the SDVRP in conjunction with column generation.

Chapter 3

A Construction Heuristic for the Split Delivery Vehicle Routing Problem

This chapter discusses construction heuristic results for the SDVRP. Section 3.1 formally describes the SDVRP including a mathematical flow formulation. The heuristic solution procedure is discussed in Section 3.2, Section 3.3 summarizes the computational experience of this procedure, and Section 3.4 summarizes the results.

3.1 Problem Statement and Flow Formulation

The SDVRP can be defined as follows. Let $G = (N, A)$ be a complete and directed graph, where $N = 1, 2, \ldots, n$ is the set of nodes and $A = \{(i, j) : i, j \in N, i \neq j\}$ is the set of arcs. Node 1 represents the depot and nodes $2, 3, \ldots, n$ represent customers. The cost of traversing an arc is denoted by c_{ij}, and it is assumed that c_{ij} is non-negative and the triangle inequality holds. The demand of each customer is denoted by d_i. The vehicles, assumed to be identical, have a capacity of Q and are based at the depot. The total demand D, where $D = \sum d_i$, and vehicle capacity are used to determine the minimum number of routes K, where $K = \lceil D/Q \rceil$. The SDVRP consists of determining K routes from the depot by minimizing the total distance traveled, while satisfying the customer demand and not exceeding vehicle capacity.

3.1.1 SDVRP Flow Formulation

The following formulation is a combination of the directed graph formulation of Belenguer et al. (2000) and the undirected graph formulation of Dror et al. (1994). This formulation assumes that c_{ij} satisfies the triangle inequality and that exactly the minimum number of vehicle routes, K, is used. The formulation does not assume that distances are symmetric.

Indexed Sets:

$i = 1, 2, \ldots, n$; node index

$j = 1, 2, \ldots, n$; node index

$k = 1, 2, \ldots, m$; route index

Parameters:

m: the number of vehicle routes

n: the number of nodes

Q: the vehicle capacity

c_{ij}: the cost or distance from node i to node j

d_i: the demand of customer i

where $d_1 = 0$.

Decision Variables:

x_{ijk}: a binary variable that is one when arc i, j is traversed on route k; zero otherwise

u_{ik}: free variable used in the subtour elimination constraints

y_{ik}: a binary variable that is one when node i is visited on route k; zero otherwise

v_{ik}: a variable that denotes the amount of material delivered to node i on route k

without loss of generality, y_{ik} and v_{ik} are not defined for $i = 1$.

Objective:

Minimize $$Z = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} c_{ij} \sum_{k=1}^{m} x_{ijk} \tag{3.1}$$

Constraints:

$$\sum_{k=1}^{m} v_{ik} = d_i, \forall\, i = 2, ..., n \tag{3.2}$$

$$\sum_{i=2}^{n} v_{ik} \leq Q, \forall\, k = 1, ..., m \tag{3.3}$$

$$\sum_{\substack{i=1 \\ i \neq p}}^{n} x_{ipk} - \sum_{\substack{j=1 \\ j \neq p}}^{n} x_{pjk} = 0, \forall\, k = 1, \ldots, m;\ p = 1, \ldots, n \tag{3.4}$$

$$u_{ik} - u_{jk} + n x_{ijk} \leq n - 1, \forall\, i = 2, ..., n;\ i \neq j;\ k = 1, ..., m \tag{3.5}$$

$$d_i y_{ik} \geq v_{ik}, \forall\, k = 1, ..., m;\ i = 2, ..., n \tag{3.6}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ijk} = y_{ik}, \forall\, k = 1, ..., m;\ i = 2, ..., n \tag{3.7}$$

$$\sum_{j=2}^{n} x_{1jk} + x_{j1k} = 2, \forall\, k = 1, ..., m \tag{3.8}$$

$$x_{ijk} \in \{0,1\}, \forall\, i = 1, \ldots, n;\ j = 1, \ldots, n;\ i \neq j;\ k = 1, \ldots, m \tag{3.9}$$

$$y_{ik} \in \{0,1\}, \forall\, i = 2, \ldots, n;\ k = 1, \ldots, m \tag{3.10}$$

$$v_{ik} \geq 0, \forall\, i = 2, \ldots, n;\ k = 1, \ldots, m \tag{3.11}$$

The objective is represented by (3.1), which is to minimize the total distance traveled. Constraints (3.2) and (3.3) guarantee that all customer demand is satisfied without violating vehicle capacity. Constraints (3.4) and (3.5) ensure flow conservation and that subtours are eliminated, respectively. Constraints (3.6) and (3.7) force the binary variables to be positive if material is delivered to node i on route k. Constraint (3.8) ensures that the depot is entered and exited on every vehicle route, and constraints (3.9) – (3.11) provide variable restrictions. Additional, mathematically redundant, constraints that can be used to possibly strengthen the formulation are stated below. These additional constraints were not included in the SDVRP flow formulation based on results from preliminary testing.

<u>Additional Constraints:</u>

$$y_{ik} \leq v_{ik}, \forall\, k = 1, ..., m;\ i = 2, ..., n \tag{3.12}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} d_i x_{ijk} \geq v_{ik}, \forall\, k = 1, ..., m;\ i = 2, ..., n \tag{3.13}$$

$$\sum_{i=2}^{n} y_{ik} \geq 1, \forall\, k = 1, ..., m \tag{3.14}$$

$$\sum_{k=1}^{m} y_{ik} \geq 1, \forall\, i = 2,...,n \tag{3.15}$$

$$x_{ijk} + x_{jik} \leq 1, \forall\, i = 2,...,n;\ j = 2,...,n;\ i \neq j;\ k = 1,...,m \tag{3.16}$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{k=1}^{m} x_{ijk} \leq n + 2m - 2 \tag{3.17}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{m} y_{ik} \leq n + m - 2 \tag{3.18}$$

$$x_{ijk} \leq \sum_{\substack{p=1 \\ p \neq i}}^{n} x_{jpk}, \forall\, k = 1,...,m;\ i = 2,...,n;\ j = 2,...,n \tag{3.19}$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ijk} \leq \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ijk+1}, \forall\, k = 1,\ldots,m-1 \tag{3.20}$$

$$\sum_{j=2}^{n} c_{1j} x_{1jk} \leq \sum_{i=2}^{n} c_{i1} x_{i1k}, \forall\, k = 1,\ldots,m \tag{3.21}$$

$$y_{ik} + y_{ip} + y_{jk} + y_{jp} \leq 3, \forall\, i = 2,...,n;\ j = 2,...,n;\ i \neq j;\ k = 1,...,m;\ p = 1,...,m;\ p \neq k \tag{3.22}$$

Constraint (3.12) guarantees that if a customer is visited then material is delivered, and constraint (3.13) does not allow material to be delivered unless a customer is visited. First introduced by Belenguer et al. (2000), constraints (3.14) and (3.15) ensure each route is used and a customer is visited at least once, respectfully. Constraint (3.16) eliminates two-node subtours. Dror et al. (1994) introduced constraint (3.17), which eliminates some k-split cycles by placing an upper-bound on the number of arcs that can be used in an optimal solution. Constraint (3.18), which follows from constraint (3.17), restricts the total number of customer visits within an optimal solution. Constraint (3.19), introduced by Dror et al. (1994), removes some types of fractional cycles when the binary restrictions of x_{ijk} are relaxed. Constraint (3.20) eliminates alternative optimal

solutions for routes by sorting the routes from shortest to longest. Alternative optimal solutions are nullified by Constraint (3.21) for the direction of the particular route by having the shortest arc traversed first of the two arcs adjacent to the depot. 2-Split deliveries are forbidden by Constraint (3.22) to nodes i and j on both routes k and p. There exist additional constraints that were not listed; however, only constraints that were beneficial in preliminary testing were noted.

3.2 Construction Heuristic Procedure

The heuristic procedure presented in this section constructs a set of feasible solutions to the SDVRP. The first procedure of the heuristic initializes the input. The second procedure assigns customers to vehicle routes, iteratively. The order of each route is then determined by a TSP solution procedure, finalizing the solution. Finally, the solution is outputted. The computer code for the heuristic assumes that customer demand and vehicle capacity are integers. Similar to the flow formulation in Section 3.1, the computer code for the heuristic does not assume symmetric distances between two nodes. Ties are broken arbitrarily, unless otherwise specified.

3.2.1 Partition Procedure

The partition procedure assigns customers and their demand to vehicle routes. Initially, the procedure checks for feasibility (i.e., adequate capacity). The iterative

process begins with Control 1, which is completed for each vehicle route. A customer, with unsatisfied demand, is selected and assigned to the vehicle route in Control 1. The capacity threshold is calculated in Control 2. If the vehicle is not loaded within the capacity threshold, then the algorithm proceeds to Control 3. Otherwise, Control 1 is repeated for the next vehicle route. Control 3 selects a number of customers closest to the set of customers already assigned to the vehicle route and determines whether or not they will be added to the route. Control 3 is repeated until the vehicle is loaded within the capacity threshold or until all customer demand is satisfied. The outputs from this procedure are the v_{ik} and y_{ik} variables. This procedure is shown in Figure **3-1**.

There are three decisions made in the partition procedure for each vehicle route, each decision corresponds to a control from Figure **3-1**. Control 1 selects an initial node to begin the vehicle route. Control 2 calculates the capacity threshold. Control 3 determines how many nodes to evaluate per iteration. This dissertation will focus on eight rules for Control 1 and three rules each for Control 2 and Control 3; thus, there are seventy-two possible combinations if every rule is completed for the three controls. There are an infinite amount of rules for each Control, but based on preliminary testing, intuition, and time constraints we have focused on a smaller set of rules.

Control 1:

The initial node for each vehicle route is selected from the set of customers with unsatisfied demand. This initial node is selected using one of the following eight rules: furthest node from the depot, closest node from the depot, median node with respect to distance from depot, node with highest demand, node with smallest demand, furthest

node on smallest arc (i.e., the arc i, j , where $i \neq j, i \neq 1, j \neq 1$, for which c_{ij} is the smallest), random node, and furthest node based on the arc from the Clarke – Wright (1964) procedure. The Clarke – Wright procedure calculates a savings $s_{ij} = c_{1i} - c_{ij} + c_{j1}$ for each arc i, j where $i \neq j, i \neq 1, j \neq 1$, and the arc with the largest savings is selected. These eight rules may select the same initial node for a particular vehicle route; however, they have the potential to be diverse in their selection.

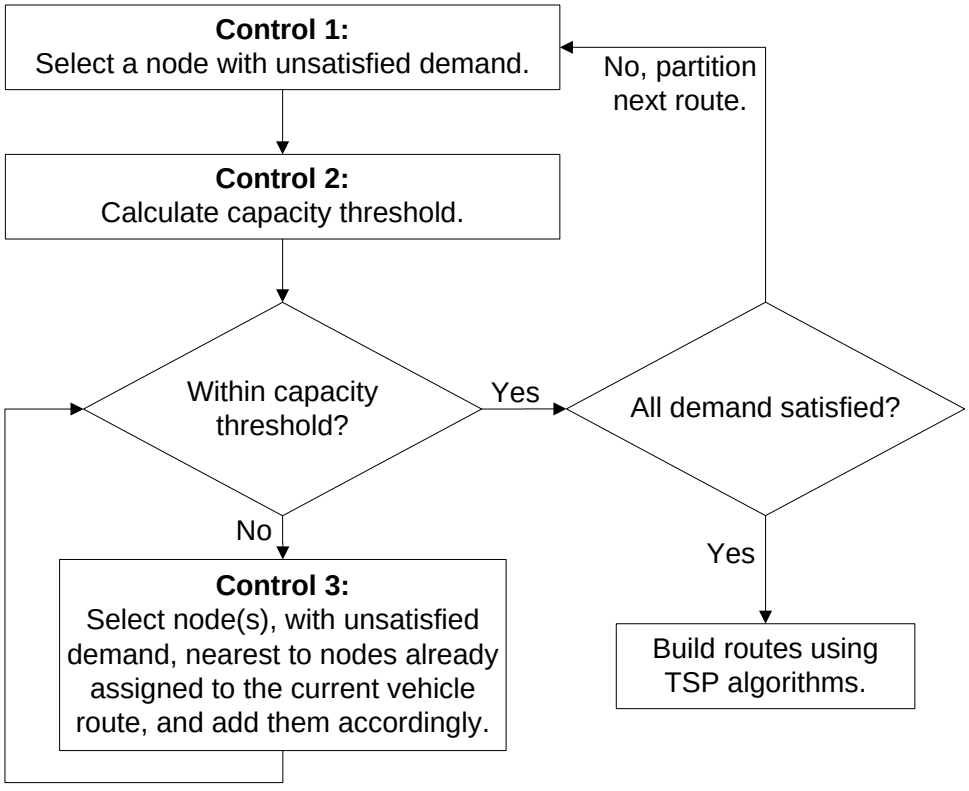

Figure **3-1**: Partition Procedure for SDVRP Construction Heuristic.

Control 2:

The capacity threshold allows the spare vehicle capacity (i.e., the total amount of spare vehicle capacity across all routes, calculated by $mQ-D$) to be shared amongst all of the vehicle routes. This threshold is calculated by three different rules. The first rule takes the average spare capacity (i.e., remaining spare vehicle capacity divided by the remaining number of vehicle routes), based on the remaining number of vehicle routes, and rounds the answer down to the nearest integer. The second rule takes the average spare capacity, based on the remaining number of vehicle routes, and rounds the answer up to the nearest integer. The third rule allows for the current vehicle route to use all of the spare capacity. These rules can be ignored if there is no spare vehicle capacity, since in that case all of the vehicles must be filled to capacity.

Control 3:

The node selection decision evaluates the closest (relative to the nodes already on the vehicle route) one, two, or three nodes per iteration. If only one node is selected, then it is evaluated to determine if all or part of its remaining demand will be placed on the route. All of the demand will be placed on the route if it fits; otherwise, the vehicle route will be filled and the node's demand will be adjusted. If two nodes are selected, then they are evaluated to determine if one or both can be placed on the vehicle route. Both nodes are added to the route if all of their remaining demand fits on the route. If, individually, both of the nodes remaining demand exceed the vehicle capacity, then only a portion of the closest node's demand is placed on the route. Otherwise, all of the closest node's demand is placed on the route and only a portion of the second closest

node's demand. If three nodes are selected, then they are evaluated to determine if one, two, or all will be added to the vehicle route.

Step-by-step Procedure Example:

The following procedure is an example of one of the 72 combinations. Control 1 takes the furthest node from the depot, Control 2 rounds the average spare capacity up to the nearest integer, and Control 3 evaluates the closest two nodes. This example is shown in Figure **3-2**. In addition to the sets and variables presented in Section 3.1, the following data and variables are used in this procedure:

S: total available spare capacity across all vehicle routes (i.e., $S = mQ - D$)

A: average spare capacity per route (i.e., $A = S/m$, rounded up to the nearest integer)

R: set of nodes with unsatisfied demand

P_k: set of nodes currently assigned to route k

f_i: amount of unsatisfied demand remaining for node i

b_k: remaining capacity of vehicle k

Step 0:

Commentary: Step 0 initializes the data and variables for the procedure.

Determine constants *m*, *n*, *D*, *Q*, *K*, and *S*. Compute the initial value of *A*.

If $m < K$, where $K = \lceil D/Q \rceil$, then STOP the problem is infeasible.

Place all nodes in set *R*, except node 1, which corresponds to the depot.

Ensure P_k is empty for all k. Set $f_i = d_i$ for all i.

Set $k = 1$.

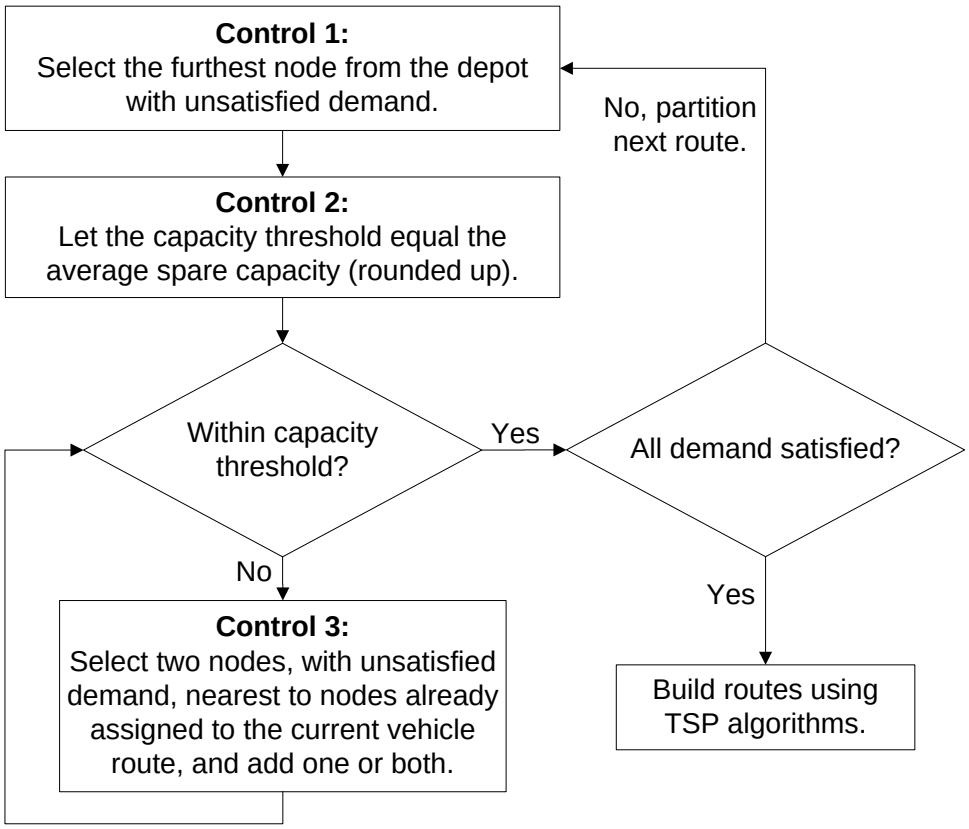

Figure **3-2**: Partition Procedure for SDVRP Construction Heuristic where Control 1 takes the furthest node from the depot, Control 2 rounds the average spare capacity up to the nearest integer, and Control 3 evaluates the closest two nodes.

Step 1:

Commentary: Step 1 selects the furthest node from the depot and adds it to the route currently being constructed, this is the first node added to the route.

If $k = m+1$, then STOP. From set *R*, find the node furthest from the depot, call it node *i*. Place node *i* in set P_k. If $f_i > Q$, then set $y_{ik} = 1$, $v_{ik} = Q$, $f_i = f_i - Q$, $A = \lceil S/(m-k) \rceil$, and $k = k + 1$; and go to Step 1. Otherwise, if $Q - f_i \leq A$, then remove *i* from set *R* and set $y_{ik} = 1$, $v_{ik} = f_i$, $S = S - Q - f_i$, $A = \lceil S/(m-k) \rceil$, $f_i = 0$, and $k = k + 1$; and go to Step 1. Otherwise, remove *i* from set *R* and set $y_{ik} = 1$, $v_{ik} = f_i$, $b_k = Q - f_i$, and $f_i = 0$; and go to Step 2.

Step 2:

Commentary: Step 2 selects nodes to add to the route currently being constructed.

If set *R* is empty, then STOP. If there is only one node in set *R*, then call it *i* and go to Step 2f. Otherwise, from set *R*, find the two closest nodes to the nodes in set P_k. Call the closest node *i* and the next closest node *j*.

If $f_i > b_k$ and $f_j \leq b_k$, then go to Step 2a.

If $f_i \leq b_k$ and $f_j > b_k$, then go to Step 2b.

If $f_i \leq b_k$, $f_j \leq b_k$, and $f_i + f_j > b_k$, then go to Step 2c.

If $f_i \leq b_k$, $f_j \leq b_k$, and $f_i + f_j \leq b_k$, then go to Step 2d.

If $f_i > b_k$ and $f_j > b_k$, then go to Step 2e.

Step 2a:

Commentary: Step 2a adds node j to the route currently being constructed, and then determines if more nodes need to be considered.

If $b_k - f_j \leq A$, then remove j from set R, place j in P_k, and set $y_{jk} = 1$, $v_{jk} = f_j$, $S = S - b_k - f_j$, $A = \lceil S/(m-k) \rceil$, $f_j = 0$, and $k = k + 1$; and go to Step 1. Otherwise, remove j from set R, place j in P_k, and set $y_{jk} = 1$, $v_{jk} = f_j$, $b_k = b_k - f_j$, and $f_j = 0$; and go to Step 2.

Step 2b:

Commentary: Step 2b adds node i to the route currently being constructed, and then determines if more nodes need to be considered.

If $b_k - f_i \leq A$, then remove i from set R, place i in P_k, and set $y_{ik} = 1$, $v_{ik} = f_i$, $S = S - b_k - f_j$, $A = \lceil S/(m-k) \rceil$, $f_i = 0$, and $k = k + 1$; and go to Step 1. Otherwise, remove i from set R, place i in P_k, and set $y_{ik} = 1$, $v_{ik} = f_i$, $b_k = b_k - f_i$, and $f_i = 0$; and go to Step 2.

Step 2c:

Commentary: Step 2c adds node i to the route currently being constructed, and then determines if more nodes need to be considered.

If $f_i = b_k$, then remove i from set R, place i in P_k, and set $y_{ik} = 1$, $v_{ik} = f_i$, $A = \lceil S/(m-k) \rceil$, $f_i = 0$, and $k = k + 1$; and go to Step 1. Otherwise, remove i from set R, place i in P_k, and set $y_{ik} = 1$, $y_{jk} = 1$, $v_{ik} = f_i$, $v_{jk} = b_k - f_i$, $A = \lceil S/(m-k) \rceil$, $f_i = 0$, $f_j = f_j - v_{jk}$, and $k = k+1$; and go to Step 1.

Step 2d:

Commentary: Step 2d adds nodes i and j to the route currently being constructed, and then determines if more nodes need to be considered.

If $b_k - f_i - f_j \leq A$, then remove i and j from set R, place i and j in P_k, and set $y_{ik} = 1$, $y_{jk} = 1$, $v_{ik} = f_i$, $v_{jk} = f_j$, $S = S - (b_k - f_i - f_j)$, $A = \lceil S/(m-k) \rceil$, $f_i = 0$, $f_j = 0$, and $k = k + 1$; and go to Step 1. Otherwise, remove i and j from set R, place i and j in P_k, and set $y_{ik} = 1$, $y_{jk} = 1$, $v_{ik} = f_i$, $v_{jk} = f_j$, $b_k = b_k - f_i - f_j$, $f_i = 0$, and $f_j = 0$; and go to Step 2.

Step 2e:

Commentary: Step 2e adds node i to the route currently being constructed, and then determines if more nodes need to be considered.

Place i in P_k and set $y_{ik} = 1$, $v_{ik} = b_k$, $f_i = f_i - b_k$, $A = \lceil S/(m-k) \rceil$, and $k = k + 1$; and go to Step 1.

Step 2f:

Commentary: Step 2f adds node i to the route currently being constructed.

Place i in P_k. If $f_i \leq b_k$, then remove i from set R and set $y_{ik} = 1$, $v_{ik} = f_i$, and STOP. Otherwise, set $y_{ik} = 1$, $v_{ik} = b_k$, $f_i = f_i - b_k$, $A = \lceil S/(m-k) \rceil$, and $k = k + 1$; and go to Step 1.

The above example is the procedure for Control 1 taking the furthest node from the depot, Control 2 rounding the average spare capacity up to the nearest integer, and

Control 3 evaluating the closest two nodes. A flowchart of this example is shown in Figure **3-2**. A simple explanation of the steps is as follows. Step 0 is initializes the problem and iteration procedure. Step 1 selects the furthest node and calculates the average spare capacity, performing Control 1 and Control 3. Step 2 selects a node or nodes to add to the vehicle route, performing Control 2. Included within each step are stopping and iterating instructions.

3.2.2 TSP Solution Procedure

The procedure presented in Section 3.2.1 assigns customers and their demand to vehicle routes. The order of each individual route is then calculated using a TSP solution procedure. If the number of customers on a particular route is small (i.e., ≤ 10), then an enumeration method ensures the best possible route with little computation. Initial solution procedures and three-opt neighborhood search procedures can be used for a larger number of customers. Syslo et al. (1983) provide detailed algorithms for the TSP, including a branch-and-bound enumeration procedure, a furthest node insertion method, and a three-opt procedure. The outputs from the TSP solution procedure are x_{ijk} variables and the objective value, Z. After the conclusion of the assignment and TSP procedures, an initial feasible solution for the SDVRP is complete.

3.3 Computational Experience of Construction Heuristic

Initial feasible solutions were constructed for the data sets from Belenguer et al. (2000) and Chen et al. (2007) based on the Construction Heuristic Procedure described in Section 3.2. The procedure was coded in FORTRAN 95 and compiled by GNU FORTRAN on an Intel Xeon Processor 2.49 GHz computer with 8 GB RAM. The best solution from the 72 combinations was outputted as the final solution for the construction heuristic for each data set. Of the 72 combinations, only nine combinations were needed to find the best solution for all data sets, and those nine combinations come from an initial set of 36 rule combinations. The best nine combinations are listed in Table **3-1**, these combinations result in the best solution for all data sets. For the computational results, the 72 combination runtime, 36 combination runtime, and the best nine runtime are reported.

The number of customers and vehicles for 11 data sets from Belenguer et al. (2000) are shown in Table **3-2**. The number of customers ranged from 50 to 100, with an additional node for the depot. The data sets also differ by amount of spare capacity per vehicle. The customers were placed randomly around a central depot and demand was generated randomly based on a high and low threshold. The number of customers and vehicles for 21 data sets from Chen et al. (2007) are shown in Table **3-3**. The number of customers ranged from eight to 288, with an additional node for the depot. The data sets do not have any spare vehicle capacity. The customers were placed on rings surrounding a central depot and the demand was either 60 or 90, with a vehicle capacity of 100.

Table **3-1**: Best nine combinations that result in the best solutions for all data sets.

Control 1	Control 2	Control 3
Farthest Node	Round-up	Closest Node
Smallest Unsatisfied Demand	Round-up	Closest Node
Farthest Node	Round-up	Closest Two Nodes
Farthest Node	Round-down	Closest Two Nodes
Largest Unsatisfied Demand	Round-up	Closest Two Nodes
Clarke-Wright	Round-up	Closest Two Nodes
Clarke-Wright	All	Closest Two Nodes
Farthest Node	Round-up	Closest Three Nodes
Clarke-Wright	Round-up	Closest Three Nodes

Table **3-2**: Eleven data sets from Belenguer et al. (2000).

Data Set	Customers	Vehicles
S51D2	50	9
S51D3	50	15
S51D4	50	27
S51D5	50	23
S51D6	50	41
S76D2	75	15
S76D3	75	23
S76D4	75	37
S101D2	100	20
S101D3	100	31
S101D5	100	48

Table **3-3**: Twenty-one data sets from Chen et al. (2007).

Data Set	Customers	Vehicles
S1	8	6
S2	16	12
S3	16	12
S4	24	18
S5	32	24
S6	32	24
S7	40	30
S8	48	36
S9	48	36
S10	64	48
S11	80	60
S12	80	60
S13	96	72
S14	120	90
S15	144	108
S16	144	108
S17	160	120
S18	160	120
S19	192	144
S20	240	180
S21	288	216

Results from Jin et al. (2007b) and Chen et al. (2007) were used as a comparison to the results from the Construction Heuristic Procedure (Section 3.2) for these 32 data sets.

Comparative results for 11 data sets from Belenguer et al. (2000) are shown in Table **3-4**. The construction heuristic (Section 3.2) produced a solution in the least amount of computer time for each data set (bold), and produced the solution with the least amount of travel distance in four cases (bold). Jin et al. (2007b) found a better solution for three data sets (bold) and Chen et al. (2007) in four cases (bold). However, Jin et al. (2007b) allowed for additional vehicles in their solution, above the minimum number required for the SDVRP, which increases the cost of the overall system.

Table **3-4**: Comparing the construction heuristic versus the two-phase method of Chen et al. (2007) and the column generation method of Jin et al. (2007b) for 11 data sets.

Data Set	Heuristic	Heuristic (72) Time (s)	Heuristic (36) Time (s)	Heuristic (9) Time (s)	Chen et al. (2007)	Time (s)	Jin et al. (2007b)	Time (s)
S51D2	735.33	27.625	10.438	**2.110**	-	-	**722.93***	10741
S51D3	998.75	2.719	1.360	**0.094**	-	-	**968.85**	833
S51D4	1647.25	0.282	0.157	**0.031**	**1586.5**	201.74	1605.64*	789
S51D5	1405.30	0.266	0.156	**0.046**	**1355.5**	201.62	1361.24*	10
S51D6	2267.08	0.125	0.063	**0.016**	2197.8	301.9	**2196.35***	478
S76D2	**1143.16**	111.109	50.532	**6.422**	-	-	1146.68*	75074
S76D3	**1471.76**	12.172	10.563	**0.312**	-	-	1474.89	3546
S76D4	2166.86	12.359	11.532	**0.141**	**2136.4**	601.92	2157.87*	369
S101D2	**1458.76**	231.938	80.453	**11.250**	-	-	1460.54*	189392
S101D3	**1945.23**	27.265	22.094	**0.703**	-	-	1956.91*	36777
S101D5	2881.35	3.516	2.297	**0.453**	**2846.2**	645.99	2885*	5043

** Jin et al. (2007b) starred-solutions used more than the minimum number of vehicles, computer specifications unavailable, and computer solution time includes time to compute both lower and upper bounds.*
Heuristic presented cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.

Comparative results for 21 data sets from Chen et al. (2007) are shown in Table **3-5**. The construction heuristic (Section 3.2) produced a solution in the least amount of computer time for each data set (bold), and produced the solution with the least amount of travel distance in 16 cases (bold). Chen et al. (2007) found a better feasible solution for four data sets (bold). Both methods found a solution with the same objective for data set S1. Chen et al. (2007) reported pseudo lower bounds based on a graphical estimation described in Chen (2007). Chen et al. (2007) finds a feasible solution that matches this pseudo lower bound for four instances (italic). The construction heuristic (Section 3.2) finds a feasible solution that matches this pseudo lower bound for five data sets (italic), and finds a feasible solution lower than this bound for nine data sets (italic and underline).

Using lower bounds from previous literature (Jin et al., 2007b) the duality gap of the construction heuristic solution is shown in Table **3-6** for 11 data sets. The duality gap varies from 5.26% to 8.72%, with a solution time of no more than 11.250 seconds. The problem size ranged from 50 to 100 customers and 9 to 48 vehicle routes, as shown in Table **3-2**. Similar results for the 21 data sets are not available since a valid lower bound is not available from previous literature.

Table **3-5**: Comparing the results of the construction heuristic versus the two-phase method of Chen et al. (2007) for 21 data sets.

Data Set	Heuristic	Heuristic (72) Time (s)	Heuristic (36) Time (s)	Heuristic (9) Time (s)	Chen et al. (2007)	Time (s)
S1	***228.28***	**< 0.001**	**< 0.001**	**< 0.001**	***228.28***	0.7
S2	***708.28***	**< 0.001**	**< 0.001**	**< 0.001**	714.4	54.4
S3	***430.58***	**< 0.001**	**< 0.001**	**< 0.001**	*430.61*	67.3
S4	640.02	0.015	**< 0.001**	**< 0.001**	***631.06***	400
S5	***1390.57***	0.031	0.016	**< 0.001**	1408.12	402.7
S6	860.46	0.032	0.015	**< 0.001**	***831.21***	408.3
S7	***3640.00***	0.062	0.032	**< 0.001**	3714.4	403.2
S8	***5068.28***	0.100	0.062	**0.015**	5200	404.1
S9	2071.05	0.125	0.047	**0.016**	**2059.84**	404.3
S10	2772.03	0.281	0.141	**0.031**	**2749.11**	400
S11	***13280.00***	0.578	0.266	**0.063**	13612.12	400.1
S12	***7279.97***	0.672	0.375	**0.078**	7399.06	408.3
S13	***10110.57***	1.094	0.641	**0.094**	10367.06	404.5
S14	***10786.52***	1.860	1.000	**0.203**	11023	5021.7
S15	**15160.04**	3.360	1.844	**0.344**	15271.77	5042.3
S16	**3434.81**	2.437	1.187	**0.313**	3449.05	5014.7
S17	***26559.92***	4.766	2.360	**0.484**	26665.76	5023.6
S18	***14302.22***	4.172	2.156	**0.500**	14546.58	5028.6
S19	***20152.53***	7.453	3.890	**0.844**	20559.21	5034.2
S20	***39706.51***	21.141	10.594	**2.312**	40408.22	5053
S21	**11461.20**	26.219	12.719	**3.313**	11491.67	5051

Heuristic presented cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.

Table **3-6**: Duality GAP for construction heuristic solution for 11 data sets.

Data Set	Linear Relaxation	Lower Bound	Construction Heuristic Objective	GAP	Heuristic (Best 9) Time (s)
S51D2	379.36	693.13	735.33	5.74%	2.110
S51D3	379.36	920.86	998.75	7.80%	0.094
S51D4	379.36	1503.54	1647.25	8.72%	0.031
S51D5	379.36	1291.70	1405.30	8.08%	0.046
S51D6	379.36	2107.85	2267.08	7.02%	0.016
S76D2	485.53	1065.26	1143.16	6.81%	6.422
S76D3	485.53	1394.36	1471.76	5.26%	0.312
S76D4	485.53	2019.91	2166.86	6.78%	0.141
S101D2	579.90	1344.35	1458.76	7.84%	11.250
S101D3	579.90	1831.57	1945.23	5.84%	0.703
S101D5	579.90	2724.32	2881.35	5.45%	0.453

Lower bounds from Jin et al. (2007b).

The solutions from the construction heuristic are not guaranteed to be local optimal solutions. A neighborhood search was implemented using two or three exchanges amongst the vehicle routes from the best solution and resolving the necessary TSPs. However, the objective value for the 32 data sets only decreased in a few cases, and then only by a small percentage, while using considerable extra computation time (i.e., much longer than the construction heuristic). Therefore, using a neighborhood search was determined to be ineffective.

3.4 Summary

This chapter focused on formulating and solving the SDVRP using a flow formulation and a construction heuristic. A flow formulation and additional constraints were presented. The primary research result of this chapter is a construction heuristic that provides good initial starting solutions (within 8.72% duality GAP for 11 data sets)

for the SDVRP in little computer time (within 11.250 seconds for all test cases). The best solution from this construction heuristic could then be used as a starting solution for other methods, such as Tabu search (Archetti et al., 2006a), column generation (Jin et al., 2007b), and the two-step method developed by Chen et al. (2007). The construction heuristic does not assume symmetric distances, and a future research direction would be to test this heuristic with asymmetric data sets.

Chapter 4

A Genetic Algorithm for the Split Delivery Vehicle Routing Problem

This chapter discusses a hybrid genetic algorithm and computational results for the SDVRP. Section 4.1 describes the hybrid genetic algorithm procedure, Section 4.2 summarizes the computational experience of this procedure, and Section 4.3 summarizes the results.

4.1 Hybrid Genetic Algorithm Procedure

A genetic algorithm is a global search procedure that solves problems by emulating evolution. A pure genetic algorithm uses reproduction and mutation to develop a new generation of solutions from the current generation of solutions. The constraints of the SDVRP do not allow the application of pure genetic algorithms without an additional step to ensure feasibility. A *hybrid* genetic algorithm allows for a genetic global search procedure while ensuring feasibility. The phrase *hybrid genetic algorithm* is sometimes used to describe *memetic algorithms*; however, for this dissertation *hybrid* refers to composing a solution from multiple sources. Coupling *hybrid* and genetic algorithms yields the term hybrid genetic algorithm. This section is organized as follows, Section 4.1.1 describes the development of an initial population and the reproduction procedure is discussed in Section 4.1.2.

4.1.1 Initial Population

The construction heuristic presented in Section 3.2 develops initial solutions for the SDVRP. If the same set of rules is applied for the three controls for each vehicle route, then up to 72 different solutions can be created. However, a more diverse set of solutions can be created if a different set of rules for each control is applied for each vehicle route. By randomly selecting which rule to apply for each specific control for each vehicle route, a feasible solution can be generated. The results of producing 100 solutions by randomly applying the rules are shown in Table **4-1** and Table **4-2**. These tables follow the same formatting and computer specifications as Table **3-4** and Table **3-5**, respectively. Only the solution with the least amount of travel distance is reported.

The 100 solutions with a random application of the rules were generated rather quickly, less than 205 seconds for any particular data set. The construction heuristic produced a better or equivalent solution than the minimum of the 100 random solutions for 29 of the 32 data sets. The three data sets where the random solution produced a smaller objective were S51D5, S4, and S10; by 0.95%, 1.40%, and 0.14%, respectively. The higher runtimes for data sets S76D2 and S101D2 are due to the larger number of stops on a particular vehicle route for some of the poorer solutions generated by a random application of the rules. Thus, for these solutions the TSP Solution Procedure, discussed in Section 3.2.2, takes longer to complete. In order to provide the hybrid genetic algorithm with a strong and diverse start, the initial population for the hybrid genetic algorithm included the 72 combination solutions and the 100 solutions generated by randomly applying the rules for each vehicle route.

Table **4-1**: The construction heuristic versus 100 random solutions for 11 data sets.

Data Set	Heuristic	Heuristic (All 72) Time (s)	Heuristic (Set 36) Time (s)	Heuristic (Best 9) Time (s)	Heuristic (Random 100)	Time (s)	% Difference
S51D2	**735.33**	27.625	10.438	**2.110**	746.48	25.671	-1.52%
S51D3	**998.75**	2.719	1.360	**0.094**	1019.77	1.890	-2.10%
S51D4	**1647.25**	0.282	0.157	**0.031**	1672.24	0.375	-1.52%
S51D5	1405.30	0.266	0.156	**0.046**	**1392.00**	0.406	**0.95%**
S51D6	**2267.08**	0.125	0.063	**0.016**	2290.74	0.187	-1.04%
S76D2	**1143.16**	111.109	50.532	**6.422**	1164.63	143.859	-1.88%
S76D3	**1471.76**	12.172	10.563	**0.312**	1544.45	7.203	-4.94%
S76D4	**2166.86**	12.359	11.532	**0.141**	2233.08	3.468	-3.06%
S101D2	**1458.76**	231.938	80.453	**11.250**	1491.51	204.781	-2.25%
S101D3	**1945.23**	27.265	22.094	**0.703**	1984.41	11.610	-2.01%
S101D5	**2881.35**	3.516	2.297	**0.453**	2943.48	3.469	-2.16%

Computer specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
*% Difference = (Heuristic Solution - Heuristic Random) / Heuristic Solution * 100.*

Table **4-2**: The construction heuristic versus 100 random solutions for 21 data sets.

Data Set	Heuristic	Heuristic (72) Time (s)	Heuristic (36) Time (s)	Heuristic (9) Time (s)	Heuristic (Random 100)	Time (s)	% Difference
S1	***228.28***	**< 0.001**	**< 0.001**	**< 0.001**	***228.28***	**< 0.001**	0.00%
S2	***708.28***	**< 0.001**	**< 0.001**	**< 0.001**	***708.28***	**< 0.001**	0.00%
S3	***430.58***	**< 0.001**	**< 0.001**	**< 0.001**	***430.58***	**< 0.001**	0.00%
S4	640.02	0.015	**< 0.001**	**< 0.001**	***631.05***	0.016	**1.40%**
S5	***1390.57***	0.031	0.016	**< 0.001**	1419.30	0.063	-2.07%
S6	**860.46**	0.032	0.015	**< 0.001**	888.18	0.047	-3.22%
S7	***3640.00***	0.062	0.032	**< 0.001**	***3640.00***	0.094	0.00%
S8	***5068.28***	0.100	0.062	**0.015**	***5068.28***	0.141	0.00%
S9	**2071.05**	0.125	0.047	**0.016**	2099.33	0.157	-1.37%
S10	2772.03	0.281	0.141	**0.031**	**2768.19**	0.343	**0.14%**
S11	***13280.00***	0.578	0.266	**0.063**	13300.00	0.703	-0.15%
S12	***7279.97***	0.672	0.375	**0.078**	7335.64	0.765	-0.76%
S13	***10110.57***	1.094	0.641	**0.094**	10184.77	1.156	-0.73%
S14	***10786.52***	1.860	1.000	**0.203**	10892.37	2.391	-0.98%
S15	**15160.04**	3.360	1.844	**0.344**	15290.55	4.140	-0.86%
S16	**3434.81**	2.437	1.187	**0.313**	3481.96	3.594	-1.37%
S17	***26559.92***	4.766	2.360	**0.484**	26703.21	5.469	-0.54%
S18	***14302.22***	4.172	2.156	**0.500**	14434.65	5.562	-0.93%
S19	***20152.53***	7.453	3.890	**0.844**	20366.07	10.015	-1.06%
S20	***39706.51***	21.141	10.594	**2.312**	39891.36	26.468	-0.47%
S21	**11461.20**	26.219	12.719	**3.313**	11555.21	37.000	-0.82%

Computer specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
*% Difference = (Heuristic Solution - Heuristic Random) / Heuristic Solution * 100.*

4.1.2 Reproduction Procedure

The 100 randomly generated solutions and the 72 solutions developed by the construction heuristic were used as the initial population. Subsequent offspring populations were created route-by-route using a hybrid genetic algorithm to ensure feasibility. A variety of parameter settings were analyzed and tuned based on the 32 data sets. The results from two fitness approaches are given, shortest route and largest demand unit per distance unit.

Fitness Approach 1: Shortest Route

In order to build a single feasible solution a number of steps must be completed. The current population of solutions provides a set of vehicle routes, and these routes are sorted from shortest to longest based on travel distance. The first fitness approach is to select shorter routes that meet a certain capacity threshold with a greater probability of being selected than longer routes. The shortest feasible route is selected with a probability P_g, and this probability is the same for all solutions and routes. If the vehicle route is selected, then it is added to the current solution and is not included in any further solutions (neither the current solution nor any future solutions) for the current population. If the route is not selected, then the next shortest feasible route is selected with probability P_g. If there are no feasible routes remaining, then the solution is completed using the construction heuristic with a random rule selection for the remaining vehicle routes. By using the construction heuristic, feasibility is ensured. In addition, a number of good solutions from the previous generation are included in the current generation, and

bad solutions generated were discarded. This is often referred to as *memory* (Louis and Li, 1997; Acan and Tekol, 2003). This procedure ensures that each generation is better than the previous, and builds a set of good solutions.

The parameters are capacity threshold, probability of route being selected P_g, and the number of solutions from the previous generation kept in the current generation. These parameters were tuned using the 32 data sets. The results of this analysis were to set the capacity threshold as the average vehicle slack (rounded up). The probability, P_g, of a route being selected was set to 20%. The number of solutions kept from the previous generation was set at 10%, which means that the worst 10% of the next generation solutions were discarded (unless they were more favorable than the best 10% from the previous generation). The population size for a generation was set at 100 solutions (except for the initial population which was 172 solutions) and the number of new generations is 20. These values were used to ensure the entire procedure was completed in a timely fashion.

Usually, genetic algorithms incorporate a mutation stage which randomly alters a small portion of the solutions from generation to generation. The SDVRP has side constraints (i.e., capacity, demand) that make mutation difficult while ensuring feasibility. Using the construction heuristic to finish building solutions where no feasible route existed provided a method to alter solutions from generation to generation. This is not a direct mutation, but this method does allow for a portion of the current population to be randomly altered while maintaining feasibility.

The final solution outputted by the hybrid genetic algorithm is the best solution from the last generation. However, it is possible to find this solution in a previous

generation, but it would have remained in the current generation since it would have been better than the worst 10% of solutions. This concept goes along with the *survival of the fittest* goal of genetic algorithms.

Fitness Approach 2: Ratio of Demand Unit Versus Distance Unit

The first fitness approach uses only distances and does not take demand into account during the selection of routes. The second fitness approach sorts the feasible routes based on demand units divided by distance units. The larger this ratio, the more likely the route is selected. The route with the largest demand unit per distance unit is selected with probability P_g. All other parameters remained the same as Fitness Approach 1.

Step-by-step Procedure for Hybrid Genetic Algorithm:

Step 0: Build the initial population using the construction heuristic and 100 random solutions by randomly applying the rules for each of the three controls. The total initial population size is 172.

Step 1: Build the current generation. Sort the routes in the previous population based on the fitness approach. Build a new solution iteratively by route by using Step 2.

Step 2: Select a feasible route with the best fitness value with probability $P_g = 0.20$. If the feasible route is selected, then add it to the current solution and discard it from being used in later solutions in the current generation. If the feasible route is not selected, then repeat Step 2 (the feasible route is not discarded, but it is not allowed to be selected during the current iteration when selecting a vehicle route).

Step 3: Repeat Step 2 until a complete solution is built or until all feasible routes have been exhausted. If all feasible routes have been exhausted, then use the construction heuristic (by randomly applying the set of rules for each control) to build the remaining routes for the solution.

Step 4: Repeat Steps 2 and 3 until the entire population of 100 solutions is built.

Step 5: Compare the worst 10% of the current population of solutions to the best solutions from the previous generation. Select the best solutions (in terms of shortest travel distance) to remain in the current generation. At most 10 solutions from the current generation will be replaced.

Step 6: Repeat Step 1 until 20 generations are completed.

Step 7: Select the best solution from the final generation. Output as final solution.

4.2 Computational Experience of the Hybrid Genetic Algorithm

The hybrid genetic algorithm was applied to the data sets from Belenguer et al. (2000) and Chen et al. (2007) based on the Hybrid Genetic Algorithm Procedure described in Section 4.1 for both fitness approaches. The procedure was coded in FORTRAN 95 and compiled by GNU FORTRAN on an Intel Xeon Processor 2.49 GHz computer with 8 GB RAM. The best solution was outputted as the final solution.

Section 4.2.1 describes the performance, with the 100 randomly generated solutions and the 72 solutions developed by the construction heuristic as the initial population, using the first fitness approach. Section 4.2.2 describes the performance, with poor solutions used as the initial population, using the first fitness approach. Section

4.2.3 describes performance with the second fitness procedure and good initial solutions, by taking the route with the largest demand unit per distance unit with probability P_g. Section 4.2.4 describes the performance of the second fitness approach with poor initial solutions.

4.2.1 Computational Experience with Good Solutions in the Initial Population (Fitness Approach One)

Comparative results for 11 data sets from Belenguer et al. (2000) are shown in Table **4-3**. The fitness approach one hybrid genetic algorithm time is separated into initial population time and reproduction time, and a total algorithm time is provided. The genetic algorithm produced a solution in the least amount of computer time for each data set (bold), except S51D5. The hybrid genetic algorithm produced the solution with the least amount of travel distance in four cases (bold) [same data sets as construction heuristic]. Jin et al. (2007b) found a better solution for three data sets (bold) and Chen et al. (2007) in four cases (bold). However, Jin et al. (2007b) allowed for additional vehicles in their solution, above the minimum number required for the SDVRP, which increases the cost of the overall system.

Comparative results for 21 data sets from Chen et al. (2007) are shown in Table **4-4**. The hybrid genetic algorithm produced a solution in the least amount of computer time for each data set (bold), except S20 and S21. The first fitness approach hybrid genetic algorithm found the solution with the least amount of travel distance in 18 cases (bold) [one additional data set, S4, compared to construction heuristic]. Chen et al.

Table **4-3**: Comparing the hybrid genetic algorithm (good start) versus the two-phase method of Chen et al. (2007) and the column generation method of Jin et al. (2007b) for 11 data sets for the first fitness approach.

Data Set	Genetic Solution	Initial Population Time (s)	Reproduction Time (s)	Total Genetic Algorithm Time (s)	Chen et al. (2007) Objective	Time (s)	Jin et al. (2007) Objective	Time (s)
S51D2	727.66^	53.093	46.125	**99.218**	-	-	**722.93***	10741
S51D3	998.75	4.609	5.922	**10.531**	-	-	**968.85**	833
S51D4	1647.25	0.641	21.296	**21.937**	**1586.5**	201.74	1605.64*	789
S51D5	1388.75^	0.672	13.312	13.984	**1355.5**	201.62	1361.24*	**10**
S51D6	2267.08	0.328	71.000	**71.328**	2197.8	301.9	**2196.35***	478
S76D2	**1143.16**	255.000	292.844	**547.844**	-	-	1146.68*	75074
S76D3	**1471.76**	19.375	22.640	**42.015**	-	-	1474.89	3546
S76D4	2166.86	15.937	55.172	**71.109**	**2136.4**	601.92	2157.87*	369
S101D2	**1458.76**	356.734	248.203	**604.937**	-	-	1460.54*	189392
S101D3	**1945.23**	38.969	46.453	**85.422**	-	-	1956.91*	36777
S101D5	2881.35	6.969	117.547	**124.516**	**2846.2**	645.99	2885*	5043

** Jin et al. (2007) starred-solutions used more than the minimum number of vehicles, computer specifications unavailable, and computer solution time includes time to compute both lower and upper bounds.*
Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.
Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
^ Genetic Algorithm improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

Table **4-4**: Comparing the results of the hybrid genetic algorithm (good start) versus the two-phase method of Chen et al. (2007) for 21 data sets for the first fitness approach.

Data Set	Genetic Solution	Pre-Genetic Time (s)	Reproduction Time (s)	Total Genetic Algorithm Time (s)	Chen et al. (2007) Objective	Time (s)
S1	***228.28***	< 0.001	0.266	**0.266**	***228.28***	0.7
S2	***708.28***	0.015	1.922	**1.937**	714.4	54.4
S3	***430.58***	< 0.001	1.937	**1.937**	*430.61*	67.3
S4	***631.05^***	0.031	6.203	**6.234**	631.06	400
S5	***1390.57***	0.094	14.406	**14.500**	1408.12	402.7
S6	860.46	0.079	14.437	**14.516**	***831.21***	408.3
S7	***3640.00***	0.157	28.109	**28.266**	3714.4	403.2
S8	***5068.28***	0.266	48.015	**48.281**	5200	404.1
S9	2071.05	0.250	48.125	**48.375**	**2059.84**	404.3
S10	2768.19^	0.593	113.672	**114.265**	**2749.11**	400
S11	***13280.00***	1.266	219.641	**220.907**	13612.12	400.1
S12	***7279.97***	1.453	219.922	**221.375**	7399.06	408.3
S13	***10110.57***	2.219	428.531	**430.750**	10367.06	404.5
S14	***10786.52***	4.250	746.922	**751.172**	11023	5021.7
S15	**15160.04**	7.500	1273.985	**1281.485**	15271.77	5042.3
S16	**3434.81**	6.016	1272.734	**1278.750**	3449.05	5014.7
S17	***26559.92***	10.235	1743.890	**1754.125**	26665.76	5023.6
S18	***14302.22***	9.750	1743.688	**1753.438**	14546.58	5028.6
S19	***20152.53***	17.500	3019.203	**3036.703**	20559.21	5034.2
S20	***39706.51***	47.547	6164.688	6212.235	40408.22	**5053**
S21	**11461.20**	63.297	10628.437	10691.734	11491.67	**5051**

Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.
Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
^ Genetic Algorithm improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

(2007) found a better feasible solution for three data sets (bold). Both methods found a solution with the same objective for data set S1. Chen et al. (2007) reported pseudo lower bounds based on a graphical estimation described in Chen (2007). Chen et al. (2007) finds a feasible solution that matches this pseudo lower bound for four instances (italic). The hybrid genetic algorithm finds a feasible solution that matches this pseudo lower bound for five data sets (italic), and finds a feasible solution lower than this bound for ten data sets (italic and underline).

The reproduction portion of the hybrid genetic algorithm produced a better solution than the initial population for only four of the 32 test cases; S51D2, S51D5, S4, and S10. However, as discussed in Section 3.3, the construction heuristic provides good starting solutions (within 8.72% duality GAP) for the 11 data sets. Therefore, there was not much room for improvement over the best solution from the initial population. Supplementary results showed that the genetic algorithm was effective in finding better solutions when given an inferior (i.e., completely random) initial population. However, the objective of the final solution using that random initial population was poor when compared to using the construction heuristic to provide a portion of the initial population.

4.2.2 Computational Experience with Poor Solutions in the Initial Population (Fitness Approach One)

Poor solutions were generated to create an initial population for the hybrid genetic algorithm. The 172 solutions for the initial population were created by using a specific set of rules with the construction heuristic by always applying the random node (Control

1), all spare capacity (Control 2), and evaluating one node at a time (Control 3). These solutions were still optimized using the TSP Procedure (Section 3.2.2); thus, they were not completely *random*.

Comparative results to the hybrid genetic algorithm with good initial solutions from Section 4.2.1 for 11 data sets from Belenguer et al. (2000) are shown in Table **4-5**. The hybrid genetic algorithm time is separated into initial population time and reproduction time, and a total genetic algorithm time is provided. The outputted solution from hybrid genetic algorithm with poor initial solutions improved the best initial solution by 1.76% to 5.78%. The hybrid genetic algorithm with poor initial solutions did not compare well to the hybrid genetic algorithm with good initial solutions when comparing solution time and solution quality.

Comparative results to the hybrid genetic algorithm with good initial solutions from Section 4.2.1 for 21 data sets from Chen et al. (2007) are shown in Table **4-6**. The hybrid genetic algorithm time is separated into initial population time and reproduction time, and a total genetic algorithm time is provided. The outputted solution from hybrid genetic algorithm with poor initial solutions improved the best initial solution by 0% (no improvement) to 3.09%. The hybrid genetic algorithm with poor initial solutions did not compare well to the hybrid genetic algorithm with good initial solutions when comparing solution time and solution quality.

Table **4-5**: Comparing the hybrid genetic algorithm (poor start) [first fitness approach] versus the hybrid genetic algorithm (good start) [first fitness approach] for 11 data sets.

Data Set	Best Initial Solution	Worst Initial Solution	Average Initial Solution	Genetic Solution (Poor Start)	% Improvement (Over Best)	% Improvement (Over Average)	Initial Population Time (s)	Reproduction Time (s)	Total Genetic Algorithm Time (s)	Genetic Solution (Good Start)	Total Time (s)
S51D2	774.84	931.21	846.58	739.24	4.59%	12.68%	60.106	48.050	108.155	**727.66^**	**99.218**
S51D3	1061.17	1223.40	1152.68	1003.12	5.47%	12.97%	4.960	7.667	12.627	**998.75**	**10.531**
S51D4	1723.68	1898.47	1785.46	1649.89	4.28%	7.59%	0.764	32.064	32.828	**1647.25**	**21.937**
S51D5	1436.90	1596.85	1521.62	1405.30	2.20%	7.64%	0.701	15.614	16.315	**1388.75^**	**13.984**
S51D6	2310.64	2515.02	2408.90	2269.92	1.76%	5.77%	0.329	111.932	112.261	**2267.08**	**71.328**
S76D2	1202.82	1343.97	1275.08	1150.74	4.33%	9.75%	285.144	540.374	825.518	**1143.16**	**547.844**
S76D3	1568.54	1754.98	1662.32	1513.41	3.52%	8.96%	20.332	43.278	63.610	**1471.76**	**42.015**
S76D4	2278.65	2533.37	2385.01	2178.92	4.38%	8.64%	17.652	95.010	112.662	**2166.86**	**71.109**
S101D2	1563.13	1729.75	1635.37	1472.82	5.78%	9.94%	386.376	302.794	689.170	**1458.76**	**604.937**
S101D3	2054.51	2265.33	2157.03	1960.04	4.60%	9.13%	42.273	49.081	91.353	**1945.23**	**85.422**
S101D5	3024.88	3216.84	3099.27	2893.50	4.34%	6.64%	7.682	168.806	176.488	**2881.35**	**124.516**

Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.

^ Genetic Algorithm (Good Start) improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

% Improvement = [Best Initial Solution - Genetic Solution (Poor Start)]/ Best Initial Solution

Table **4-6**: Comparing the hybrid genetic algorithm (poor start) [first fitness approach] versus the hybrid genetic algorithm (good start) [first fitness approach] for 21 data sets.

Data Set	Best Initial Solution	Worst Initial Solution	Average Initial Solution	Genetic Solution (Poor Start)	% Improvement (Over Best)	% Improvement (Over Average)	Initial Population Time (s)	Reproduction Time (s)	Total Genetic Algorithm Time (s)	Genetic Solution (Good Start)	Total Time (s)
S1	228.28	320.64	262.28	228.28	0.00%	12.96%	< 0.001	0.319	0.319	**228.28**	**0.266**
S2	708.28	912.53	809.36	708.28	0.00%	12.49%	0.016	2.064	2.080	**708.28**	**1.937**
S3	435.32	573.73	489.08	431.59	0.86%	11.75%	0.001	2.198	2.199	**430.58**	**1.937**
S4	650.97	790.05	714.20	648.45	0.39%	9.21%	0.034	6.738	6.772	**631.05^**	**6.234**
S5	1425.25	1691.98	1558.04	1412.84	0.87%	9.32%	0.100	16.216	16.316	**1390.57**	**14.500**
S6	887.62	981.66	931.76	882.51	0.57%	5.29%	0.080	15.818	15.899	**860.46**	**14.516**
S7	3640.00	4193.71	3875.40	3640.00	0.00%	6.07%	0.161	32.745	32.906	**3640.00**	**28.266**
S8	5126.50	5806.86	5476.57	5123.96	0.05%	6.44%	0.273	56.131	56.404	**5068.28**	**48.281**
S9	2163.14	2481.22	2259.50	2120.92	1.95%	6.13%	0.264	56.894	57.158	**2071.05**	**48.375**
S10	2842.70	3091.62	2974.53	2835.55	0.25%	4.67%	0.621	126.647	127.268	**2768.19^**	**114.265**
S11	13573.60	14431.17	13983.54	13502.29	0.53%	3.44%	1.267	232.756	234.023	**13280.00**	**220.907**
S12	7516.61	8152.04	7750.70	7499.99	0.22%	3.23%	1.527	232.680	234.206	**7279.97**	**221.375**
S13	10472.36	11103.86	10785.25	10239.74	2.22%	5.06%	2.353	503.158	505.512	**10110.57**	**430.750**
S14	11172.88	11802.30	11456.53	11046.02	1.14%	3.58%	4.335	870.623	874.959	**10786.52**	**751.172**
S15	15524.16	16334.41	15965.99	15422.44	0.66%	3.40%	7.629	1369.703	1377.332	**15160.04**	**1281.485**
S16	3474.56	3598.84	3525.35	3461.70	0.37%	1.81%	6.040	1377.044	1383.084	**3434.81**	**1278.750**
S17	27417.59	28417.93	27899.55	26652.80	2.79%	4.47%	10.536	1767.638	1778.175	**26559.92**	**1754.125**
S18	14858.16	15420.44	15162.35	14398.93	3.09%	5.03%	10.455	2072.349	2082.803	**14302.22**	**1753.438**
S19	20905.76	21624.20	21188.41	20507.17	1.91%	3.22%	18.048	3335.910	3353.958	**20152.53**	**3036.703**
S20	40861.41	42044.93	41456.12	39875.19	2.41%	3.81%	48.960	6496.007	6544.967	**39706.51**	**6212.235**
S21	11600.32	11854.52	11706.46	11477.71	1.06%	1.95%	66.038	10644.505	10710.543	**11461.20**	**10691.734**

Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.

^ Genetic Algorithm (Good Start) improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

% Improvement (Over Best) = [Best Initial Solution - Genetic Solution (Poor Start)]/ Best Initial Solution

4.2.3 Computational Experience with Good Solutions in the Initial Population (Fitness Approach Two)

Comparative results for 11 data sets from Belenguer et al. (2000) are shown in Table **4-7**. The second fitness approach hybrid genetic algorithm time is separated into initial population time and reproduction time, and a total genetic algorithm time is provided. Both fitness approaches produced the solution with the least amount of travel distance in four cases (bold) [same data sets as construction heuristic]. Jin et al. (2007b) found a better solution for three data sets (bold) and Chen et al. (2007) in four cases (bold). However, Jin et al. (2007b) allowed for additional vehicles in their solution, above the minimum number required for the SDVRP, which increases the cost of the overall system. The second fitness approach improved the initial population solution in seven of the 11 test cases; whereas, the first fitness approach only improved in two of the 11 test cases. The second fitness approach also found a better solution, when compared to the first approach, in six of the 11 test cases, they tied in four cases (S76D2, S76D3, S101D2, S101D3), and the first approach found a better solution in only one data set (S51D2). The four cases in which they tied, neither approach improved upon the best solution provided in the initial population, and these were the four data sets in which the hybrid genetic algorithm outperformed Jin et al. (2007b) and Chen et al. (2007).

Comparative results for 21 data sets from Chen et al. (2007) are shown in Table **4-8**. The second fitness approach found the solution with the least amount of travel distance in 19 cases (bold) [three additional data sets, S4, S9, and S10, compared to

Table **4-7**: Comparing the hybrid genetic algorithm (second fitness approach) versus the two-phase method of Chen et al. (2007) and the column generation method of Jin et al. (2007b) for 11 data sets.

Data Set	Genetic Solution (Second Approach)	Initial Population Time (s)	Reproduction Time (s)	Total Second Approach Time (s)	Genetic Solution (First Approach)	Total First Approach Time (s)	Chen et al. (2007) Objective	Time (s)	Jin et al. (2007) Objective	Time (s)
S51D2	730.87^	53.093	46.064	**99.157**	727.66^	99.218	-	-	**722.93***	10741
S51D3	994.80^	4.609	5.785	**10.394**	998.75	10.531	-	-	**968.85**	833
S51D4	1637.47^	0.641	21.468	22.109	1647.25	**21.937**	**1586.5**	201.74	1605.64*	789
S51D5	1385.08^	0.672	13.266	13.938	1388.75^	13.984	**1355.5**	201.62	1361.24*	**10**
S51D6	2229.64^	0.328	74.122	74.450	2267.08	**71.328**	2197.8	301.9	**2196.35***	478
S76D2	**1143.16**	255.000	284.721	**539.721**	**1143.16**	547.844	-	-	1146.68*	75074
S76D3	**1471.76**	19.375	22.145	**41.520**	**1471.76**	42.015	-	-	1474.89	3546
S76D4	2165.25^	15.937	53.572	**69.509**	2166.86	71.109	**2136.4**	601.92	2157.87*	369
S101D2	**1458.76**	356.734	248.894	605.628	**1458.76**	**604.937**	-	-	1460.54*	189392
S101D3	**1945.23**	38.969	45.486	**84.455**	**1945.23**	85.422	-	-	1956.91*	36777
S101D5	2873.08^	6.969	120.897	127.866	2881.35	**124.516**	**2846.2**	645.99	2885*	5043

** Jin et al. (2007) starred-solutions used more than the minimum number of vehicles, computer specifications unavailable, and computer solution time includes time to compute both lower and upper bounds.*
Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.
Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
^ Genetic Algorithm improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

Table **4-8**: Comparing the hybrid genetic algorithm (second fitness approach) versus the two-phase method of Chen et al. (2007) for 21 data sets.

Data Set	Genetic Solution (Second Approach)	Initial Population Time (s)	Reproduction Time (s)	Total Second Approach Time (s)	Genetic Solution (First Approach)	Total First Approach Time (s)	Chen et al. (2007) Objective	Time (s)
S1	***228.28***	< 0.001	0.272	0.272	***228.28***	**0.266**	***228.28***	0.7
S2	***708.28***	0.015	1.935	1.950	***708.28***	**1.937**	714.4	54.4
S3	***430.58***	< 0.001	1.939	1.939	***430.58***	**1.937**	*430.61*	67.3
S4	***631.05^***	0.031	6.207	6.238	***631.05^***	**6.234**	631.06	400
S5	***1390.57***	0.094	14.104	**14.198**	***1390.57***	14.500	1408.12	402.7
S6	833.58^	0.079	14.887	14.966	860.46	**14.516**	***831.21***	408.3
S7	***3640.00***	0.157	28.453	28.610	***3640.00***	**28.266**	3714.4	403.2
S8	***5068.28***	0.266	47.994	**48.260**	***5068.28***	48.281	5200	404.1
S9	**2054.84^**	0.250	48.655	48.905	2071.05	**48.375**	2059.84	404.3
S10	**2746.54^**	0.593	113.564	**114.157**	2768.19^	114.265	2749.11	400
S11	***13280.00***	1.266	230.377	231.643	***13280.00***	**220.907**	13612.12	400.1
S12	***7279.97***	1.453	225.656	227.109	***7279.97***	**221.375**	7399.06	408.3
S13	***10110.57***	2.219	419.732	**421.951**	***10110.57***	430.750	10367.06	404.5
S14	***10786.52***	4.250	714.396	**718.646**	***10786.52***	751.172	11023	5021.7
S15	**15160.04**	7.500	1270.850	**1278.350**	**15160.04**	1281.485	15271.77	5042.3
S16	**3433.83^**	6.016	1219.865	**1225.881**	**3434.81**	1278.750	3449.05	5014.7
S17	***26559.92***	10.235	1711.962	**1722.197**	***26559.92***	1754.125	26665.76	5023.6
S18	***14302.22***	9.750	1726.084	**1735.834**	***14302.22***	1753.438	14546.58	5028.6
S19	***20152.53***	17.500	3075.671	3093.171	***20152.53***	**3036.703**	20559.21	5034.2
S20	***39706.51***	47.547	6160.615	6208.162	***39706.51***	6212.235	40408.22	**5053**
S21	**11461.20**	63.297	10502.406	10565.703	**11461.20**	10691.734	11491.67	**5051**

Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.
Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
^ Genetic Algorithm improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

construction heuristic]. Chen et al. (2007) found a better feasible solution for one data set (bold) [S6]. Both methods found a solution with the same objective for data set S1. Chen et al. (2007) reported pseudo lower bounds based on a graphical estimation described in Chen (2007). Chen et al. (2007) finds a feasible solution that matches this pseudo lower bound for four instances (italic). The hybrid genetic algorithm (both fitness approaches) finds a feasible solution that matches this pseudo lower bound for five data sets (italic), and finds a feasible solution lower than this bound for ten data sets (italic and underline). When comparing the two fitness approaches, neither is consistently faster than the other in reproduction time. However, the second fitness approach finds a better solution in three cases (S6, S9, S10), and the second fitness approach improves the initial population's best solution in five cases (S4, S6, S9, S10, S16), whereas the first fitness approach improves the initial solution in only two cases (S4, S10).

Based on the comparison between the two fitness approaches (with good initial solutions), neither approach seems to be faster than the other. Based on the 32 data sets the two methods were always within 5% of each other in reproduction runtime. However, the second fitness approach provides a better improvement in most cases, with the only exception being S51D2. The two approaches tie in many cases where neither improves upon the best solution from the initial population.

4.2.4 Computational Experience with Poor Solutions in the Initial Population (Fitness Approach Two)

Poor solutions were generated to create an initial population for the second fitness approach for the hybrid genetic algorithm. The initial population was created in the same way as described in Section 4.2.2.

Comparative results to the second fitness approach for the hybrid genetic algorithm with good initial solutions from Section 4.2.3 for 11 data sets from Belenguer et al. (2000) are shown in Table **4-9**. The hybrid genetic algorithm time is separated into initial population time and reproduction time, and a total genetic algorithm time is provided. The outputted solution from the hybrid genetic algorithm with poor initial solutions improved the best initial solution by 1.82% to 6.06%. The hybrid genetic algorithm with poor initial solutions (second fitness approach) did not compare well to the hybrid genetic algorithm with good initial solutions (second fitness approach) when comparing solution time and solution quality.

Comparative results to the hybrid genetic algorithm with good initial solutions from Section 4.2.1 for 21 data sets from Chen et al. (2007) are shown in Table **4-10**. The hybrid genetic algorithm time is separated into initial population time and reproduction time, and a total genetic algorithm time is provided. The outputted solution from the hybrid genetic algorithm with poor initial solutions improved the best initial solution by 0% (no improvement) to 2.98%. The hybrid genetic algorithm with poor initial solutions (second fitness approach) did not compare well to the hybrid genetic algorithm with good

Table **4-9**: Comparing the hybrid genetic algorithm (poor start) [second fitness approach] versus the hybrid genetic algorithm (good start) [second fitness approach] for 11 data sets.

Data Set	Best Initial Solution	Worst Initial Solution	Average Initial Solution	Genetic Solution (Poor Start)	% Improvement (Over Best)	% Improvement (Over Average)	Initial Population Time (s)	Reproduction Time (s)	Total Genetic Algorithm Time (s)	Genetic Solution (Good Start)	Total Time (s)
S51D2	774.84	931.21	846.58	732.87	5.42%	13.43%	60.106	48.259	108.365	**730.87^**	**99.157**
S51D3	1061.17	1223.40	1152.68	996.90	6.06%	13.51%	4.960	7.562	12.522	**994.80^**	**10.394**
S51D4	1723.68	1898.47	1785.46	1647.20	4.44%	7.74%	0.764	31.351	32.116	**1637.47^**	**22.109**
S51D5	1436.90	1596.85	1521.62	1396.27	2.83%	8.24%	0.701	15.290	15.992	**1385.08^**	**13.938**
S51D6	2310.64	2515.02	2408.90	2268.52	1.82%	5.83%	0.329	110.902	111.231	**2229.64^**	**74.450**
S76D2	1202.82	1343.97	1275.08	1163.33	3.28%	8.76%	285.144	562.863	848.007	**1143.16**	**539.721**
S76D3	1568.54	1754.98	1662.32	1499.54	4.40%	9.79%	20.332	42.912	63.244	**1471.76**	**41.520**
S76D4	2278.65	2533.37	2385.01	2173.23	4.63%	8.88%	17.652	98.245	115.897	**2165.25^**	**69.509**
S101D2	1563.13	1729.75	1635.37	1484.72	5.02%	9.21%	386.376	290.825	677.200	**1458.76**	**605.628**
S101D3	2054.51	2265.33	2157.03	1959.94	4.60%	9.14%	42.273	48.937	91.209	**1945.23**	**84.455**
S101D5	3024.88	3216.84	3099.27	2886.72	4.57%	6.86%	7.682	174.228	181.910	**2873.08^**	**127.866**

Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.

^ Genetic Algorithm (Good Start) improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

% Improvement = [Best Initial Solution - Genetic Solution (Poor Start)]/ Best Initial Solution

Table **4-10**: Comparing the hybrid genetic algorithm (poor start) [second fitness approach] versus the hybrid genetic algorithm (good start) [second fitness approach] for 21 data sets.

Data Set	Best Initial Solution	Worst Initial Solution	Average Initial Solution	Genetic Solution (Poor Start)	% Improvement (Over Best)	% Improvement (Over Average)	Initial Population Time (s)	Reproduction Time (s)	Total Genetic Algorithm Time (s)	Genetic Solution (Good Start)	Total Time (s)
S1	228.28	320.64	262.28	228.28	0.00%	12.96%	< 0.001	0.320	0.320	**228.28**	**0.272**
S2	708.28	912.53	809.36	708.28	0.00%	12.49%	0.016	2.050	2.066	**708.28**	**1.950**
S3	435.32	573.73	489.08	432.51	0.64%	11.56%	0.001	2.184	2.185	**430.58**	**1.939**
S4	650.97	790.05	714.20	647.35	0.56%	9.36%	0.034	7.007	7.041	**631.05^**	**6.238**
S5	1425.25	1691.98	1558.04	1405.56	1.38%	9.79%	0.100	16.707	16.806	**1390.57**	**14.198**
S6	887.62	981.66	931.76	882.43	0.58%	5.29%	0.080	16.371	16.452	**833.58^**	**14.966**
S7	3640.00	4193.71	3875.40	3640.00	0.00%	6.07%	0.161	32.395	32.556	**3640.00**	**28.610**
S8	5126.50	5806.86	5476.57	5098.04	0.56%	6.91%	0.273	54.445	54.718	**5068.28**	**48.260**
S9	2163.14	2481.22	2259.50	2117.21	2.12%	6.30%	0.264	56.504	56.768	**2054.84^**	**48.905**
S10	2842.70	3091.62	2974.53	2826.21	0.58%	4.99%	0.621	131.008	131.629	**2746.54^**	**114.157**
S11	13573.60	14431.17	13983.54	13489.67	0.62%	3.53%	1.267	238.847	240.114	**13280.00**	**231.643**
S12	7516.61	8152.04	7750.70	7435.01	1.09%	4.07%	1.527	239.478	241.004	**7279.97**	**227.109**
S13	10472.36	11103.86	10785.25	10203.92	2.56%	5.39%	2.353	501.350	503.704	**10110.57**	**421.951**
S14	11172.88	11802.30	11456.53	10960.60	1.90%	4.33%	4.335	874.636	878.971	**10786.52**	**718.646**
S15	15524.16	16334.41	15965.99	15524.16	0.00%	2.77%	7.629	1303.062	1310.691	**15160.04**	**1278.350**
S16	3474.56	3598.84	3525.35	3466.41	0.23%	1.67%	6.040	1430.784	1436.824	**3433.83^**	**1225.881**
S17	27417.59	28417.93	27899.55	26601.27	2.98%	4.65%	10.536	1728.077	1738.613	**26559.92**	**1722.197**
S18	14858.16	15420.44	15162.35	14418.97	2.96%	4.90%	10.455	2054.735	2065.189	**14302.22**	**1735.834**
S19	20905.76	21624.20	21188.41	20453.36	2.16%	3.47%	18.048	3209.622	3227.671	**20152.53**	**3093.171**
S20	40861.41	42044.93	41456.12	40053.71	1.98%	3.38%	48.960	6658.730	6707.690	**39706.51**	**6208.162**
S21	11600.32	11854.52	11706.46	11437.43	1.40%	2.30%	66.038	12045.051	12111.090	**11461.20**	**10565.703**

Genetic Algoirthm cpu specifications: FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.

^ Genetic Algorithm (Good Start) improves upon the Initial Population from the Construction Heuristic and 100 Random Solutions.

% Improvement (Over Best) = [Best Initial Solution - Genetic Solution (Poor Start)]/ Best Initial Solution

initial solutions (second fitness approach) when comparing solution time and solution quality.

4.3 Summary

This chapter focused on formulating and solving the SDVRP using a hybrid genetic algorithm. The primary research result of this chapter is a hybrid genetic algorithm procedure that provides comparable solutions based on objective value and computer time for the SDVRP when compared to a column generation procedure (Jin et al., 2007b) and a two-step method (Chen et al., 2007). Of the two fitness approaches, the second fitness approach performed better for most of the 32 data sets in terms of solution quality. Neither fitness approach was better than the other in solution time. As with the construction heuristic, the hybrid genetic algorithm does not assume symmetric distances, and a future research direction would be to test this heuristic with asymmetric data sets.

The performance of the hybrid genetic algorithm is affected by the relative strength of the construction heuristic, which is used as an input to the hybrid genetic algorithm. This was shown by testing the hybrid genetic algorithm, both first and second fitness approaches, with randomly generated solutions which were still optimized by using the TSP Solution Procedure (Section 3.2.2). The results indicated that the hybrid genetic algorithm did improve the poor initial solutions, but the solution quality of the outputted solution of the genetic algorithm was better (for all data sets, both fitness approaches) when good solutions were used in the initial population.

A novel extension to the hybrid genetic algorithm would be to use the information provided by the population to aid in producing new solutions. For example, suppose the current population consists of a set of solutions and a pairwise analysis is done for the nodes. For each pair of nodes *i* and *j*, π_{ij} is computed, where π_{ij} is the probability of the population (complete set of vehicle routes) where *i* and *j* happen to lie on the same vehicle route. Also, c_{ij}^{Y} is the average objective value of those solutions where *i* and *j* do lie on the same route and c_{ij}^{N} is the average objective value when *i* and *j* do not lie on the same route. Then, systematically use π_{ij}, c_{ij}^{Y}, and c_{ij}^{N} to determine whether *i* and *j* should be on the same vehicle route in the next generation. For example, given $i = 2$ and $j = 9$, let $\pi_{29} = 0.9$ and $c_{29}^{Y} << c_{29}^{N}$, then nodes $i = 2$ and $j = 9$ would have a high probability of being placed on the same route in the next generation.

Chapter 5

Set-covering Formulation and Column Generation Procedure for the Split Delivery Vehicle Routing Problem

This chapter discusses a set-covering formulation and a column generation procedure for the SDVRP. Section 5.1 formally describes the SDVRP including a mathematical set-covering formulation. A proposition is discussed in Section 5.2. A lower bound comparison of the flow formulation (Section 3.1) and the set-covering formulation is given in Section 5.3. The procedure for solving the set-covering formulation using column generation is discussed and results are given in Section 5.4, and a chapter summary is provided in Section 5.5. *Note: shadow prices, dual prices, and dual variables are considered synonyms for the purposes of this Chapter.*

5.1 Problem Statement and Set-Covering Formulation

Let R denote the set of feasible routes, where c_r is the cost of route $r \in R$, and α_{ir} equals one if node i is on route r. The set of routes must be enumerated before the problem is formulated and α_{ir} must be given. The following SDVRP set-covering formulation is similar to the CVRP set-covering formulation described by Bramel and Simchi-Levi (2002). This formulation assumes that every customer's demand is less than

the vehicle capacity. However, it can be modified to allow for customer demand to exceed vehicle capacity by making y_r an integer variable instead of a binary variable.

Indexed Sets:

$i \in 1,2,\ldots n$; node index

$r \in R$; route index

Parameters:

m : the number of vehicle routes

n : the number of nodes

Q : the vehicle capacity

c_r : the cost of route r

α_{ir} : a binary parameter that is one when customer i is on route r

where $\alpha_{ir} = 0$ when $i = 1$.

d_i : the demand of customer i

where $d_1 = 0$.

Decision Variables:

y_r : a binary variable that is one when route r is used in the solution

v_{ir} : a variable that denotes the amount of material delivered to node i on route r

without loss of generality, v_{ir} is not defined for $i = 1$.

Objective:

Minimize $$Z = \sum_{r \in R} c_r y_r \tag{5.1}$$

Constraints:

$$\sum_{r \in R} v_{ir} = d_i, \forall\, i = 2, ..., n \tag{5.2}$$

$$\sum_{i=2}^{n} v_{ir} \le Q\,, \forall\, r \in R \tag{5.3}$$

$$\sum_{r \in R} y_r = m \tag{5.4}$$

$$\alpha_{ir} d_i y_r \ge v_{ir}, \forall\, i = 2, ..., n; r \in R\,; \alpha_{ir} = 1 \tag{5.5}$$

$$y_r \in \ 0,1\ , \forall\, r \in R \tag{5.6}$$

$$0 \le v_{ir} \le \alpha_{ir} d_i, \forall\, i = 2, \ldots, n; r \in R \tag{5.7}$$

The objective is represented by (5.1), which is to minimize the total distance traveled. Constraints (5.2) and (5.3) guarantee that all customer demand is satisfied without violating the vehicle capacity. Constraint (5.4) ensures the minimum number of vehicles is used. Constraint (5.4) may be omitted if there is an unlimited supply of vehicles. Constraint (5.5) forces the binary variables to be positive if material is delivered to node *i* on route *r*, and constraints (5.6) – (5.7) provide variable restrictions.

5.2 Proposition Regarding the Set-covering Formulation

In general, the set of routes, denoted by *R*, must include all possible permutations of the customer set. The total number of routes is the number of permutations of nonempty subsets with $n-1$ or fewer terms, given by: $|R| = \sum_{i=1}^{n-1} \frac{(n-1)!}{n-1-i\ !}$. Recall, $n-1$ is the total number of customers, since the depot is included in the node set. The number of non-dominated solutions is the sum of the combinations with $n-1$ or fewer terms, and is given by: $|R| = \sum_{i=1}^{n-1} \frac{(n-1)!}{i!\ n-1-i\ !}$. This requires a TSP to be solved for the set of customers assigned to route *r*.

There exists an upper-limit on the number of stops in all routes for a particular optimal solution to the SDVRP. This upper-limit is a function of the number of nodes and the minimum number of vehicle routes. Archetti and Speranza (2008) and Archetti et al. (2006) show that if the triangle inequality exists, then there exists an optimal solution to the SDVRP where the number of splits is less than the number of routes. This leads to the following proposition:

Proposition 5-1: There exists an upper-limit on the number of stops along a particular vehicle route in a particular optimal solution to the SDVRP. This upper-limit is a function of the number of nodes and their respective demands. Therefore, the number of routes in set *R* is limited by the routes that satisfy the upper-limit on the number of stops.

Discussion of Proposition 5-1:

An integer program can be created to determine the maximum number of positive x_{ijk} variables in a vehicle route by modifying the flow formulation (Section 3.1). The objective has been changed to determine the maximum number of positive variables in a vehicle route, while eliminating split deliveries. The following formulation is obtained.

Indexed Sets:

$i = 1, 2, \ldots, n$; node index

$j = 1, 2, \ldots, n$; node index

$k = 1, 2, \ldots, m$; route index

N : set of all nodes

S : any nonempty subset of $N \setminus 1$

$\overline{S}$: complement of S

Parameters:

m : the number of vehicle routes

n : the number of nodes

Q : the vehicle capacity

c_{ij} : the cost or distance from node i to node j

d_i: the demand of customer i

where $d_1 = 0$.

Decision Variables:

x_{ijk}: a binary variable that is one when arc i, j is traversed on route k; zero otherwise

u_{ik}: free variable used in the subtour elimination constraints

y_{ik}: a binary variable that is one when node i is visited on route k; zero otherwise

v_{ik}: a variable that denotes the amount of material delivered to node i on route k

w: the minimum number of stops along a particular vehicle route (not including depot stops)

without loss of generality, y_{ik} and v_{ik} are not defined for $i = 1$.

Objective:

Maximize $Z = w$ (5.8)

Constraints:

$$\sum_{k=1}^{m} v_{ik} = d_i, \forall\, i = 2, ..., n \tag{5.9}$$

$$\sum_{i=2}^{n} v_{ik} \le Q, \forall\, k = 1, ..., m \tag{5.10}$$

$$\sum_{\substack{i=1 \\ i \ne p}}^{n} x_{ipk} - \sum_{\substack{j=1 \\ j \ne p}}^{n} x_{pjk} = 0, \forall\, k = 1, \ldots, m;\ p = 1, \ldots, n \tag{5.11}$$

$$u_{ik} - u_{jk} + nx_{ijk} \leq n-1, \forall\, i = 2,...,n; i \neq j; k = 1,...,m \tag{5.12}$$

$$d_i y_{ik} \geq v_{ik}, \forall\, k = 1,...,m; i = 2,...,n \tag{5.13}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ijk} = y_{ik}, \forall\, k = 1,...,m; i = 2,...,n \tag{5.14}$$

$$\sum_{j=2}^{n} x_{1jk} + x_{j1k} = 2, \forall\, k = 1,...,m \tag{5.15}$$

$$y_{ik} + y_{ip} + y_{jk} + y_{jp} \leq 3, \forall\, i = 2,...,n;\ j = 2,...,n; i \neq j; k = 1,...,m;\ p = 1,...,m;\ p \neq k \tag{5.16}$$

$$\sum_{i \in S,\ j \in \bar{S}} x_{ijk} \geq \frac{\left(\sum_{i,j \in S} x_{ijk} \right)}{|S| - 1},\ S \subseteq N \setminus 0\ ; |S| \geq 2; k = 1,...,m \tag{5.17}$$

$$x_{ijk} \leq \sum_{p \neq i} x_{jpk}, \forall\, i = 2,...,n;\ j = 2,...,n; k = 1,...,m \tag{5.18}$$

$$\sum_{i=2}^{n} \sum_{j=2}^{n} x_{ijk} \geq w, \forall\, k = 1,...,m \tag{5.19}$$

$$x_{ijk} \in \{0,1\}, \forall\, i = 1,\ldots,n;\ j = 1,\ldots,n; i \neq j; k = 1,\ldots,m \tag{5.20}$$

$$y_{ik} \in \{0,1\}, \forall\, i = 2,\ldots,n; k = 1,\ldots,m \tag{5.21}$$

$$v_{ik} \geq 0, \forall\, i = 2,\ldots,n; k = 1,\ldots,m \tag{5.22}$$

$$w \in I^{+} \tag{5.23}$$

The objective is represented by (5.8), to maximize the minimum number of positive variables in a vehicle route, denoted by w. Constraints (5.9) and (5.10) guarantee that all customer demand is satisfied without violating vehicle capacity. Constraints (5.11) and (5.12) ensure flow conservation and that subtours are eliminated,

respectively. Constraints (5.13) and (5.14) force the binary variables to be positive if material is delivered to node i on route k. Constraint (5.15) ensures that the depot is entered and exited on every vehicle route, and constraints (5.20) – (5.23) provide variable restrictions. Constraints (5.16) eliminate 2-split deliveries to nodes i and j on both routes k and p. Similar constraint sets can be added to eliminate 3-split deliveries, 4-split deliveries, etc. Constraints (5.17) and (5.18) eliminate fractional cycles as shown in Dror et al. (1994). Constraints (5.19) ensure that w is greater than or equal to the number of stops for any particular vehicle route. Combining (5.8) and (5.19) ensures the objective maximizes the minimum number of stops along a particular vehicle route. Solving the formulation for a particular data set will give an appropriate limit on *R*. The formulation provides an appropriate limit on *R*; however, this does not prove *Proposition 5-1* in a closed-form solution. Thus *Proposition 5-1* was not directly used in the column generation procedure. If a closed-form solution were developed for *Proposition 5-1*, then the column generation procedure discussed in Section 5.4 could be strengthened, by reducing the number of columns that would need to be generated to solve the problem.

5.3 Linear Programming Relaxation Comparison

The linear programming relaxations for the flow formulation (Section 3.1) and the set-covering formulation (Section 5.1) were analyzed for ten data sets. The set of non-dominated routes, *R*, was enumerated for the set-covering formulation using only three or fewer terms, the total number of routes is given by: $|R| = \sum_{i=1}^{3} \frac{(n-1)!}{i!\ n-1-i\ !}$. By definition

for the set-covering formulation (Section 5.1), the size of the set of routes, *R*, is equal to the number of binary variables in the set-covering formulation. Only three or fewer terms were included since *Proposition 5-1* was shown to be true for these ten data sets by using the formulation from Section 5.2. None of the best solutions generated from the construction heuristic had more than two customers on any vehicle route for these ten data sets, which indicates that these solutions could be optimal.

The results for the two formulations (5.1) – (5.7) versus (5.8) – (5.23) are shown in Table **5-1**, including the number of binary variables, number of constraints, and the linear relaxation. The formulation represented by (5.8) – (5.23) was solved to determine $Z^* = w$. The optimal objective value for the ten data sets was $Z^* = w = 3$. Thus, eliminating all routes except those denoted by $|R| = \sum_{i=1}^{3} \frac{(n-1)!}{i!\ n-1-i\ !}$. Computer time for the (5.8) – (5.23) formulation is also provided. The construction heuristic solution and runtime are also reported (for Best 9 Scenario). The results indicate that the set-covering formulation provides a tighter linear relaxation for the SDVRP (bold). These results indicate that solving the SDVRP with the set-covering formulation using column generation may be a promising technique.

5.4 Column Generation Procedure and Computational Results

The column generation procedure is based on the set-covering formulation (Section 5.1) and the construction heuristic (Section 3.2). The approach uses the routes generated by the construction heuristic as initial columns and then uses column

Table **5-1**: Comparing the SDVRP flow formulation and SDVRP set-covering formulation using the linear programming relaxation.

Data Set	Flow Formulation			Set-Covering Formulation				Heuristic Solution	GAP	Heuristic Time (sec)
	Binary Variables	Constraints	Linear Relaxation	Binary Variables	Constraints	Linear Relaxation	Solution Time (sec)			
S1	480	658	80.00	92	333	**200.000**	1.086	228.284	12.39%	< 0.001
S2	3456	4100	160.00	696	2649	**549.152**	3.543	708.284	22.47%	< 0.001
S3	3456	4100	159.99	696	2649	**381.201**	3.073	430.582	11.47%	< 0.001
S4	11232	12630	186.35	2324	8997	**533.166**	10.832	640.015	16.69%	< 0.001
S5	26112	28552	319.99	5488	21425	**1063.810**	65.111	1390.568	23.50%	< 0.001
S6	26112	28552	187.24	5488	21425	**680.337**	67.540	860.461	20.93%	< 0.001
S7	50400	54170	400.00	10700	41981	**2306.346**	187.247	3640.000	36.64%	< 0.001
S8	86400	91788	480.00	18472	72713	**3137.008**	564.358	5068.284	38.11%	0.015
S9	86400	91788	426.34	18472	72713	**1526.017**	576.535	2071.051	26.32%	0.016
S10	202752	212240	507.22	43744	172897	**1971.937**	239.500	2772.034	28.86%	0.031

Proposition 5-1 Formulation Optimal Solution Objective Value (Z=w) = 3 for All Data Sets (S1 - S10)
Solution Time (sec) for the Set-Covering Formulation refers to solving the Proposition 5-1 Formulation to obtain (Z=w) = 3

generation to optimize the problem. Additional routes can be constructed as needed using a branch-and-price approach at each node of the branch-and-bound tree, Barnhart et al. (1998) describes this branch-and-price approach in detail. This method will produce a duality gap to the SDVRP, including both upper (feasible) and lower (dual) bounds.

Recall m is the number of vehicle routes and n is the number of nodes (depot and customers). Based on Archetti and Speranza (2008) and Archetti et al. (2006) the maximum number of splits is $m-1$. The minimum number of positive x_{ijk} variables (without splits) is $m+n-1$, when considering an arc to connect the depot for every vehicle route. Thus, the maximum number of positive x_{ijk} variables is $n+2m-2$.

Given that the maximum number of splits is less than the number of vehicle routes, then for each node that has a single split delivery, assign the node to one vehicle route and assign the split to the other vehicle route. If split amongst three vehicle routes, then assign the node to one vehicle route and assign the node's split to the other two vehicle routes. Then the number of "assigned splits" is less than the number of vehicle routes. Thus there exists an assignment where there is only one assigned split per vehicle route. This concept is utilized in the construction heuristic (Chapter 3) since all of the remaining demand from a node is assigned to the vehicle route before adding another node; thus, at most one "assigned split" will occur on a single vehicle route for solutions constructed through this heuristic.

The column generation procedure uses the fact that the maximum number of positive x_{ijk} variables is $n+2m-2$ to reduce the number of vehicle routes that must be

examined during the search process (i.e., to keep the size of the branch-and-bound tree small).

Branch-and-Price Approach:

The branch-and-price approach employed here is based on Barnhart et al. (1998). This approach leaves columns out of the *restricted master problem* because there are too many columns to handle efficiently. Furthermore, most of the variables will equal zero in an optimal solution of the *overall problem* and *restricted master problem*, as nonbasic variables. To verify the optimality of the *restricted master problem* solution to the *overall problem* solution a *subproblem* is completed. This *subproblem* is called the pricing problem by Barnhart et al. (1998). The *subproblem* is a separation problem for the dual variables of the *restricted master problem*, and is solved to identify appropriate columns to enter the basis. If such columns are found the *restricted master problem* is reoptimized. Branching occurs when no columns are identified to enter the basis and the *restricted master problem* solution does not satisfy the integrality conditions (as a linear programming relaxation of an integer programming problem). Branch-and-price is a generalization of branch-and-bound with linear programming relaxations, and it permits column generation to be applied throughout the branch-and-bound tree.

Barnhart et al. (1998) note that branch-and-price solution techniques for the *subproblem* are difficult because of two primary reasons. First, fixing integer variables may not be effective because it may devastate the structure of the pricing problem. Second, solving the restricted master problem (which may be a linearly relaxed version of

the overall problem) may not be efficient for a specific problem; which will lead to different rules for the branch-and-price tree for specific problem types.

If the solution to the *restricted master problem* is not integer feasible, then the branch-and-price tree evaluated until it is fathomed, yielding an optimal solution. This solution is then passed to the *subproblem*. This procedure is necessary to improve the dual bound (lower bound for SDVRP since it is a minimization problem).

Section 5.4.1 provides a step-by-step approach for the column generation procedure. Section 5.4.2 provides the computational results for the column generation procedure.

5.4.1 Step-by-step approach for the column generation procedure

The column generation procedure uses the routes generated by the construction heuristic as initial columns and then uses column generation to optimize the problem. Additional routes can be constructed as needed using a branch-and-price approach at each node of the branch-and-bound tree. This method will produce a duality gap to the SDVRP, including both upper (feasible) and lower (dual) bounds. The column generation procedure uses the fact that the maximum number of positive x_{ijk} variables is $n+2m-2$ to reduce the number of vehicle routes that must be examined during the search process (i.e., to keep the size of the branch-and-bound tree small). The following step-by-step approach is similar to the one described in Section 2.3. This step-by-step column generation procedure is depicted in Figure **5-1** following its description.

Step-by-step Column Generation Approach:

Step 0: Establish initial columns using the construction heuristic (Section 3.2).

Step 1: Solve the *restricted master problem* for the selected columns. The *restricted master problem* is the linear programming relaxed version of the formulation from Section 5.1, represented by Equations (5.1) to (5.7) with equations (5.6) linearly relaxed to $0 \le y_r \le 1, \forall\, r \in R$. Establish the branch-and-bound tree to branch on any variables that do not have an integer-feasible solution. Based on preliminary computational experience, neither of the traditional branch-and-bound methods, depth-first nor breadth-first, proved to be superior in terms of computation time. This is likely due to only one or two variables having a fractional solution for any iteration. For purposes of this dissertation, the CPLEX optimization software default branching rule was used. Once the tree is fathomed, select the optimal integer solution, then proceed to Step 2.

Step 2: Determine the optimal dual variables for the *restricted master problem*. These optimal dual variables are found by fixing the integer variables in the optimal solution from the *restricted master problem* found Step 1 and solving the corresponding linear program. The CPLEX command for fixing variables is *change problem fixed*, and the linear program optimization command is *primopt* (ILOG, 2007). Let J equal the number of basic variables and C equal the number of constraints, then let $\pi_1^J, \pi_2^J, \ldots, \pi_C^J$ be the optimal dual prices to the *restricted master problem*. Go to Step 3.

Step 3: Use the reduced costs to solve the *subproblem*. Let *j* denote the basic variable index, let *i* denote the constraint index, then the reduced costs, v^J, are solved using the formula $v^J = \underset{1 \le j \le |R|}{Minimize} \left\{ c_j - \sum_{i=1}^{C} \pi_i^J a_{ij} \right\}$. If $v^J < 0$, then the corresponding variable needs to be added to the *restricted master problem*. These variables correspond to routes, $r \in R$, where the reduced cost is negative. Thus the *subproblem* identifies routes, which need to be added to the *restricted master problem*.

These routes are identified by analyzing a combination of nodes (to form a route) that has not been considered by the *restricted master problem*. A set of columns can be established using the triangle inequality, while avoiding *k*-split cycles, and enforcing the number of total stops to be less than or equal to $n+2m-2$. This set of potential columns is then evaluated using the TSP Solution Procedure (Section 3.2.2) to identify the correct route and order of stops (from node to node); which would correspond to one variable, *r*, and thus one column.

The TSP Solution Procedure (Section 3.2.2) ensures that for a specific combination of customer nodes that a route is optimal. For example, given 1 represents the depot note, if route 1-2-3-4-1 is represented by a column in the solution, then route 1-4-2-3-1 does not need to be analyzed since the TSP procedure would have already eliminated 1-4-2-3-1 when selecting 1-2-3-4-1.

Then proceed to Step 4.

Step 4: Add the columns with negative reduced costs to the *restricted master problem*, go to Step 1. If no such variables exist (i.e., all variables have an integer solution and no negative reduced costs), then continue to Step 5.

Step 5: Report the solution, and STOP.

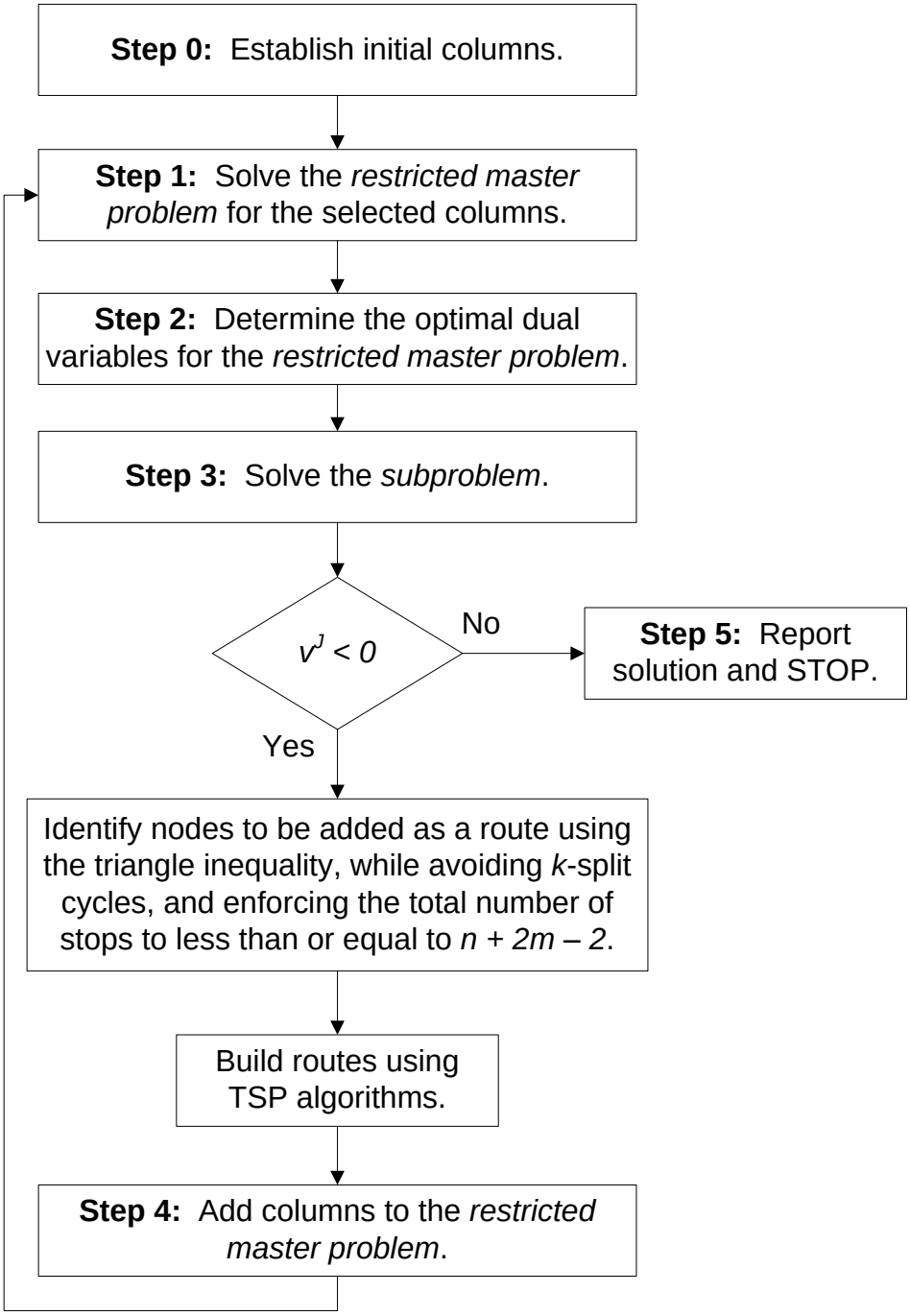

Figure **5-1**: Column Generation Procedure for SDVRP.

5.4.2 Column Generation Computation Results

The column generation procedure was applied to the data sets from Belenguer et al. (2000) and Chen et al. (2007). The procedure was coded in FORTRAN 95, compiled by GNU FORTRAN, and used in conjunction with CPLEX 11.0 on an Intel Xeon Processor 2.49 GHz computer with 8 GB RAM.

Comparative results for 11 data sets from Belenguer et al. (2000) are shown in Table **5-2**. The column generation approach, with the initial columns provided by the construction heuristic (All 72 Combinations) compared favorably to Chen et al. (2007) and Jin et al. (2007b). The column generation procedure found the best solution for 9 out of the 11 data sets (bold). The column generation procedure also produced a solution in the least amount of computer time in 9 of the 11 data sets (S51D5 and S76D4 being the exceptions). Chen et al. (2007) found the best solution for 4 out of the 11 data sets, tying the column generation procedure in all four instances (data sets S51D4, S51D5, S76D4, and S101D5). Jin et al. (2007b) found the best solution for 3 out of the 11 data sets, tying the column generation procedure in two instances (data sets S51D3 and S51D6), while finding the best solution for S51D2. Jin et al. (2007b) produced a solution in the least amount of computer time in 2 of the 11 data sets (S51D5 and S76D4). However, Jin et al. (2007b) allowed for additional vehicles in their solution, above the minimum number required for the SDVRP, which increases the cost of the overall system.

Table **5-2**: Comparing the column generation procedure to Chen et al. (2007) and Jin et al. (2007b) for 11 data sets.

Data Set	Column Generation Objective	Column Generation Dual Bound	GAP	Time (s)	Chen et al. (2007) Objective	Time (s)	Jin et al. (2007) Objective	Jin et al. (2007) Dual Bound	Time (s)
S51D2	723.75	688.83	4.83%	**1800.823**	-	-	**722.93***	**693.13**	10741
S51D3	**968.85**	920.58	4.98%	**87.150**	-	-	**968.85**	**920.86**	833
S51D4	**1586.46**	**1520.71**	4.14%	**14.804**	**1586.5**	201.74	1605.64*	1503.54	789
S51D5	**1355.47**	**1310.12**	3.35%	19.042	**1355.5**	201.62	1361.24*	1291.70	**10**
S51D6	2197.79	**2115.20**	3.76%	**9.075**	2197.8	301.9	**2196.35***	2107.85	478
S76D2	**1143.16**	**1093.39**	4.35%	**4881.883**	-	-	1146.68*	1065.26	75074
S76D3	**1471.76**	**1399.37**	4.92%	**622.935**	-	-	1474.89	1394.36	3546
S76D4	**2136.41**	**2039.11**	4.55%	791.104	**2136.4**	601.92	2157.87*	2019.91	**369**
S101D2	**1458.76**	**1395.25**	4.35%	**5101.699**	-	-	1460.54*	1344.35	189392
S101D3	**1945.23**	**1859.36**	4.41%	**1082.294**	-	-	1956.91*	1831.57	36777
S101D5	**2846.24**	2704.63	4.98%	**230.216**	**2846.2**	645.99	2885*	**2724.32**	5043

** Jin et al. (2007) starred-solutions used more than the minimum number of vehicles, computer specifications unavailable, and computer solution time includes time to compute both lower and upper bounds.*

Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.

Column Generation cpu specifications: CPLEX and FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.

Column Generation stopping criteria: 5% GAP [i.e., GAP = (Primal Solution - Dual Bound) / Primal Solution].

Column Generation time includes time to create 72 initial solutions using the construction heuristic.

Comparative results for 21 data sets from Chen et al. (2007) are shown in Table **5-3**. The column generation approach, with the initial columns provided by the construction heuristic (All 72 Combinations) compared favorably to Chen et al. (2007). The column generation procedure found the best solution for all data sets (bold), tying Chen et al. (2007) for data set S1 and S6. The column generation procedure also produced a solution in the least amount of computer time in 18 of the 21 data sets (S11, S12, and S13 being the exceptions).

For all 32 data sets the column generation procedure found a solution within 5% of optimality in 5101.699 seconds or less. The solution time includes the time to generate the initial population of 72 combinations using the construction heuristic and to maintain lower bounds by using the column generation procedure.

Table **5-3**: Comparing the column generation procedure to Chen et al. (2007) for 21 data sets.

Data Set	Column Generation Objective	Column Generation Dual Bound	GAP	Time (s)	Chen et al. (2007) Objective	Time (s)
S1	**228.28**	228.28	0.00%	**0.601**	**228.28**	0.7
S2	**708.28**	708.28	0.00%	**11.619**	714.4	54.4
S3	**430.58**	430.58	0.00%	**11.656**	430.61	67.3
S4	**631.05**	631.05	0.00%	**31.117**	631.06	400
S5	**1390.57**	1390.57	0.00%	**62.895**	1408.12	402.7
S6	**831.21**	831.21	0.00%	**62.874**	**831.21**	408.3
S7	**3640.00**	3640.00	0.00%	**162.990**	3714.4	403.2
S8	**5068.28**	5068.28	0.00%	**231.589**	5200	404.1
S9	**2050.43**	2044.23	0.30%	**231.743**	2059.84	404.3
S10	**2698.47**	2684.84	0.50%	**358.713**	2749.11	400
S11	**13280.00**	13265.29	0.11%	417.187	13612.12	**400.1**
S12	**7279.97**	7275.97	0.05%	419.324	7399.06	**408.3**
S13	**10110.57**	10093.72	0.17%	500.118	10367.06	**404.5**
S14	**10786.52**	10632.67	1.43%	**1854.501**	11023	5021.7
S15	**15160.04**	15146.92	0.09%	**3134.824**	15271.77	5042.3
S16	**3430.81**	3375.95	1.60%	**3137.551**	3449.05	5014.7
S17	**26559.92**	25320.09	4.67%	**4033.067**	26665.76	5023.6
S18	**14302.22**	14253.94	0.34%	**4033.117**	14546.58	5028.6
S19	**20152.53**	19768.23	1.91%	**4452.923**	20559.21	5034.2
S20	**39706.51**	38071.58	4.12%	**4865.231**	40408.22	5053
S21	**11461.20**	11062.32	3.48%	**4904.439**	11491.67	5051

Chen et al. (2007) cpu specifications: Visual Studio C++, CPLEX 9.0, Intel Pentium 4, 1.7 GHz, 512 MB RAM.
Column Generation cpu specifications: CPLEX and FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.
Column Generation stopping criteria: 5% GAP [i.e., GAP = (Primal Solution - Dual Bound) / Primal Solution].

5.5 Summary

This chapter focused on formulating and solving the SDVRP using a set-covering formulation and a column generation procedure. The set-covering formulation was shown to be a tighter formulation than the flow formulation presented in Section 3.1. A proposition was discussed that showed that the number of variables in the set-covering formulation can be controlled by limiting the number of stops along a vehicle route. A column generation procedure was developed and results were presented. This procedure compared favorably against two previous methods in both solution quality and computation time. The column generation procedure did not assume symmetric distances, only that the triangle inequality was satisfied.

Chapter 6

Conclusions and Future Research Directions

The SDVRP is a variant of the traditional CVRP. The SDVRP allows customers to be serviced by more than one vehicle route, which usually reduces the number of routes and the total distance traveled. This dissertation developed, for the SDVRP, a construction heuristic to create feasible solutions, a hybrid genetic algorithm, a set-covering formulation, and a column generation procedure. Computational experience was provided using data sets from previous literature. The primary contributions and future directions of this dissertation are outlined below.

6.1 Summary of Thesis Contributions

Flow Formulation:

A mathematical flow formulation for the SDVRP, with additional redundant constraints was provided in Section 3.1. This formulation does not assume symmetric distances between nodes.

Construction Heuristic:

A construction heuristic for the SDVRP was described in Section 3.2 that does not assume symmetric distances. This construction heuristic outputs good initial starting solutions (within 8.72% duality GAP for 11 data sets) for the SDVRP in little computer

time (within 11.250 seconds for all test cases). Detailed computational experience is summarized in Section 3.3.

Genetic Algorithm:

A hybrid genetic algorithm for the SDVRP was described in Section 4.1. The hybrid genetic algorithm procedure provides comparable solutions based on objective value and computer time for the SDVRP when compared to two published results, a column generation procedure (Jin et al., 2007b) and a two-step method (Chen et al., 2007). Detailed computational experience is summarized in Section 4.2. As with the construction heuristic, the hybrid genetic algorithm does not assume symmetric distances.

Set-covering Formulation:

A mathematical set-covering formulation for the SDVRP was provided in Section 5.1 that does not assume symmetric distances. A proposition was discussed in Section 5.2 on the number of vehicle stops was presented, with the conclusion being the number of total stops must be less than or equal to $n+2m-2$. It was shown in Section 5.3 that the Set-covering Formulation is much tighter than the Flow Formulation, and consequently yielded better lower bounds.

Column Generation:

Using the set-covering formulation and initial solutions from the construction heuristic a column generation procedure was presented in Section 5.4 for the SDVRP. For all 32 data sets the column generation procedure found a solution within 5% of

optimality in 5101.699 seconds or less. The solution time includes the time to generate the initial population of 72 combinations using the construction heuristic and to maintain lower bounds by using the column generation procedure. These results compare favorably against previously published results, a column generation procedure (Jin et al., 2007b) and a two-step method (Chen et al., 2007). Computational experience is summarized in Section 5.4. As with the construction heuristic and the hybrid genetic algorithm, the column generation procedure does not assume symmetric distances.

Overall Contributions:

Coupling the construction heuristic with the column generation procedure yielded the best results, in terms of objective function value. This is notable because both upper and lower bounds were generated. The construction heuristic is relatively strong at finding feasible solutions (upper bounds) in a short amount of computer time; whereas, the column generation procedure improves upon those constructed solutions and gives lower bounds. The three solution procedures did not assume symmetric distances, only that the triangle inequality was valid and that the data was deterministic for the data set.

6.2 Future Research Directions

Evaluating the SDVRP for asymmetric distances and stochastic data (i.e., customer demand, vehicle capacity, and travel distance) would be a novel extension to this dissertation, perhaps using robust optimization (Yao et al., 2009). As indicated in Section 4.3, an alternative hybrid genetic algorithm could be explored to include

information provided by the population to aid in producing new solutions. Incorporating a construction heuristic to work within a column generation procedure to create additional routes (i.e., columns) would be another extension to explore.

REFERENCES

Acan, A. and Tekol, Y., "Chromosome Reuse in Genetic Algorithms," *Lecture Notes in Computer Science*, GECCO 2003, LNCS 2723, pp. 695–705, Springer-Verlag Berlin Heidelberg 2003

Alvarenga, G.B., Mateus, G.R., and de Tomi, G., "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows," *Computers & Operations Research*, Vol. 34, pp. 1561-1584, 2007.

Archetti, C., Savelsbergh, M., and Speranza, M.G., "To split or not to split: That is the question," *Transportation Research Part E*, Vol. 44, pp. 114 – 123, 2008.

Archetti, C., Savelsbergh, M., and Hertz, A., "A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem," *Transportation Science*, Vol. 40, No. 1, pp. 64-73, 2006a.

Archetti, C., Savelsbergh, M., and Speranza, M.G., "Worst-Case Analysis for Split Delivery Vehicle Routing Problems," *Transportation Science*, Vol. 40, No. 2, pp. 226-234, 2006b.

Archetti, C. and Speranza, M.G., "The Split Delivery Vehicle Routing Problem: A Survey," *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan, E. Wasil (Editors), Springer, pp. 103-122, 2008.

Baker, B.M., and Ayechew, M.A., "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, Vol. 30, pp. 787-800, 2003.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., and Vance, P.H., "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, Vol. 46, pp. 316–329, 1998.

Belenguer, J.M., M.C. Martinez, and Mota, E., "A Lower Bound for the Split Delivery VRP," *Operations Research*, Vol. 48, No. 5, pp. 801-810, 2000.

Bradley, S.P., Hax, A.C., and Magnanti, T.L., *Applied Mathematical Programming*, Addison-Wesley, 1977.

Bramel, J. and Simchi-Levi, D., "Set-Covering-Based Algorithms for the Capacitated VRP," Toth, P. and Vigo, D. (Editors), *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics, Philadelphia, pp. 85-108, 2002.

Burrows, W., "The Vehicle Routing Problem with Loadsplitting: A Heuristic Approach," *24th Annual Conference of the Operational Research Society of New Zealand*, pp. 33-38, 1988.

Chen, S., "A Study of Four Network Problems in Transportation, Telecommunications, and Supply Chain Management," Dissertation Thesis, University of Maryland, 2007.

Chen, S., B. Golden, and Wasil, E., "The Split Delivery Vehicle Routing Problem: Applications, Test Problems, and Computational Results," *Networks*, Vol. 94, No. 4, pp. 318-329, 2007.

Clarke, G. and Wright, J., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, Vol. 12, No. 4, pp. 568-581, 1964.

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

Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.S., "New heuristics for the vehicle routing problem," *Logistics systems: Design and optimization*, A. Langevin and D. Riopel (Editors), Springer, pp. 270-297, 2005.

Dantzig, G.B. and Ramser, J.H., "The Truck Dispatching Problem," Management Science, Vol. 6, No. 1, pp. 80-91, 1959.

Desaulniers, G., Desrosiers, J., Solomon, M.M., (Editors), *Column Generation*, Springer, New York, 2005.

Dror, M., and Trudeau, P., "Savings by split delivery routing," *Transportation Science*, Vol. 23, pp. 141-145, 1989.

Dror, M., and Trudeau, P., "Split Delivery Routing," *Naval Research Logistics*, Vol. 37, pp. 383-402, 1990.

Dror, M., Laporte, G., and Trudeau, P., "Vehicle routing with split deliveries," *Discrete Applied Mathematics*, Vol. 50, pp. 239-254, 1994.

Frizzell, P.W., and Giffin, J.W., "The Split Delivery Vehicle Scheduling Problem With Time Windows and Grid Network Distances," *Computers and Operations Research*, Vol. 22, No. 6, pp. 655-667, 1995.

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

Gendreau, M., Laporte, G., and Potvin, J.-Y., "Vehicle routing: modern heuristics," Aarts, E. and Lenstra, J.K. (Editors), *Local Search in Combinatorial Optimization*, John Wiley & Sons, Inc., New York, pp. 323-330, 1997.

Gendreau, M., Laporte, G., and Potvin, J.-Y., "Metaheuristics for the Capacitated VRP," Toth, P. and Vigo, D. (Editors), *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics, Philadelphia, pp. 140-144, 2002.

Gendreau, M., Feillet, D., Dejax, P., and Gueguen, C., "Vehicle routing with time windows and split deliveries," Technical paper 2006-851, Laboratoire d'Informatique d'Avignon, 2006.

Gilmore, P., and Gomory, R., "A Linear Programming Approach to the Cutting Stock Problem," *Operations Research*, Vol. 9, pp. 849-859, 1961.

GNU GENERAL PUBLIC LICENSE, Free Software Foundation, Inc., http://fsf.org/.

Goldberg, D. *Genetic Algorithms in Search Optimization and Machine Learning*. Reading, MA, Addison Wesley, 1989.

Golden, B. L. and Assad, A. A., (Editors), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam, 1988.

Golden, B. L., Raghavan, S., and Wasil, E., (Editors), *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, 2008.

Gulczynski, D. J., Golden, B., and Wasil, E., "Recent Developments in Modeling and Solving the Split Delivery Vehicle Routing Problem, " *Tutorials in Operations Research*, INFORMS 2008, pp. 170-180, 2008.

ILOG, ILOG CPLEX 11.0 User's Manual, ILOG Optimization Documentation, 2007.

Jeon, G., Leep, H.R., and Shim, J.Y., "A vehicle routing problem solved by using a hybrid genetic algorithm," *Computers & Industrial Engineering*, Vol. 53, pp. 680-692, 2007.

Jin, M., Liu, K., and Bowden, R.O., "A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem," *International Journal of Production Economics*, Vol. 105, pp. 228-242, 2007a.

Jin, M., Liu, K., and Eksioglu, B., "A Column Generation Algorithm for the Vehicle Routing Problem with Split Delivery," *Proceedings of the 2007 Industrial Engineering Research Conference*, 2007b.

Jin, M., Liu, K., and Eksioglu, B., "A column generation approach for the split delivery vehicle routing problem," *Operations Research Letters*, pp. 265-270, 2008.

Laporte, G. and Osman, I. H., "Routing problems: A bibliography," *Annals of Operations Research*, Vol. 61, pp. 227-262, 1995.

Lee, C., Epelman, M.A., White, C.C., and Bozer, Y.A., "A shortest path approach to the multiple-vehicle routing problem with split pick-ups," *Transportation Research Part B*, Vol. 40, pp. 265-284, 2006.

Liu, K., "A Study on the Split Delivery Vehicle Routing Problem," Dissertation Thesis, Mississippi State University, 2005.

Louis, S.J. and Li, G., "Augmenting Genetic Algorithms with Memory to Solve Traveling Salesman Problems," Proceedings of the Joint Conference on Information Sciences, 1997.

Mullaseril, P., Dror, M., and Leung, J., "Split-delivery routeing heuristics in livestock feed distribution," *Journal of the Operational Research Society*, Vol. 48, pp. 107-116, 1997.

Nowak, M.A., "The pickup and delivery problem with split loads," Dissertation Thesis, Georgia Institute of Technology, 2005.

Sierksma, G. and Tijssen, G.A., "Routing helicopters for crew exchanges on off-shore locations," *Annals of Operations Research*, Vol. 76, pp. 261-286, 1998.

Song, S., Lee, K., and Kim, G., "A practical approach to solving a newspaper logistics problem using a digital map," *Computers and Industrial Engineering*, Vol. 43, pp. 315-330, 2002.

Syslo, M.M., Deo, N., and Kowalik, J.S., *Discrete Optimization Algorithms with PASCAL programs*, Prentice-Hall, 1983.

Toth, P. and Vigo, D. (Editors), *The Vehicle Routing Problem*, Society for Industrial and Applied Mathematics, Philadelphia, 2002a.

Toth, P. and Vigo, D., "Models, relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics*, Vol. 123, pp. 487-512, 2002b.

Vanderbeck, F. and Wolsey, L.A., "An exact algorithm for IP column generation," *Operations Research Letters*, Vol. 19, pp. 151-159, 1996.

Wang, X., Golden, B.L., and Wasil, E.A., "Using a Genetic Algorithm to Solve the Generalized Orienteering Problem," *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan, E. Wasil (Editors), Springer, pp. 263-274, 2008.

Winston, W.L. and Venkataramanan, M., *Introduction to Mathematical Programming, Operations Research: Volume One*, 4th Ed, Thomson Learning, 2003.

Yao, T., Mandala, S.R., and Chung, B.D., "Evacuation Transportation Planning Under Uncertainty: A Robust Optimization Approach," *Journal of Networks and Spatial Economics*, Vol. 9, Num. 2, pp. 171-189, 2009.

VITA

Joseph Hubert Wilck, IV

Personal:
Born: Wednesday; December 22, 1982 in Farmville, Virginia
Baptized: Sunday; August 19, 1990 at Farmville Baptist Church in Farmville, Virginia
Married: Saturday; July 15, 2006 to Karen R. Wilck in Bruington, Virginia

Education:
Ph.D., Industrial Engineering and Operations Research, Pennsylvania State University, 2009
M.S., Industrial and Systems Engineering, Virginia Tech, 2005
B.S., Industrial and Systems Engineering, Virginia Tech, 2004

Experience:
Assistant Professor, Industrial Engineering, University of Tennessee, August 2008 - Present
Graduate Student Lecturer, Pennsylvania State University, January 2007 - May 2008
Graduate Teaching Assistant, Pennsylvania State University, August 2005 - December 2006
Graduate Research Assistant, Virginia Tech, August 2004 - August 2005
Undergraduate Research Assistant, Virginia Tech, February 2004 - August 2004
Resident Advisor, Virginia Tech, August 2002 - May 2004
Quality Engineering Intern, Philip Morris (Richmond, VA), Summer 2003
Industrial Engineering Intern, Merck (Wilson, NC), Summer 2001 and Summer 2002

Memberships:
Institute for Operations Research and the Management Sciences (INFORMS)
Institute of Industrial Engineers (IIE)
American Society for Engineering Education (ASEE)
Member of Tau Beta Pi, The Engineering Honor Society
Member of Alpha Pi Mu, The Industrial Engineering Honor Society

Research Papers (Submitted):
Kimberly Ellis, Russell Meller, Joseph Wilck, Pratik Parikh, and Franky Marchand. "Effective Material Flow for Assembly Operations at Volvo Trucks."

Joseph Wilck and Tom Cavalier. "A Construction Heuristic for the Split Delivery Vehicle Routing Problem."

Research Papers (In Preparation):
Joseph Wilck, Steven Mills, and Marc McDill. "Computational Comparison of Stand-centered versus Cover-constraint Formulations."

Joseph Wilck and Tom Cavalier. "A Genetic Algorithm for the Split Delivery Vehicle Routing Problem."

Joseph Wilck and Tom Cavalier. "A Set-Covering Formulation and Column-Generation Procedure for the Split Delivery Vehicle Routing Problem."