INTEGRATED ROUTING MODELS FOR ENHANCED PRODUCT AND SERVICE DELIVERY

Mohammad Reihaneh

INTEGRATED ROUTING MODELS FOR ENHANCED PRODUCT AND SERVICE DELIVERY

A Dissertation Presented

by

MOHAMMAD REIHANEH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2018

Isenberg School of Management

INTEGRATED ROUTING MODELS FOR ENHANCED PRODUCT AND SERVICE DELIVERY

A Dissertation Presented

by

MOHAMMAD REIHANEH

Approved as to style and content by:

Ahmed Ghoniem, Chair

Agha Iqbal Ali, Member

Senay Solak, Member

Hari Balasubramanian, Member

George R. Milne, Department Chair
Isenberg School of Management

ABSTRACT

INTEGRATED ROUTING MODELS FOR ENHANCED PRODUCT AND SERVICE DELIVERY

SEPTEMBER 2018

MOHAMMAD REIHANEH

B.Sc., APPLIED MATH, FERDOWSI UNIVERSITY OF MASHHAD

M.Sc., IE, ISFAHAN UNIVERSITY OF TECHNOLOGY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Ahmed Ghoniem

Logistics constitutes a key function of modern-day supply chains and an indispensable prerequisite for the support and growth of conventional brick-and-mortar and online businesses. Whether for procurement or delivery purposes, manufacturers and service providers seek efficient and reliable logistical services. A 2014 Bloomberg survey reports that 73% of supply chain managers are experiencing a shift in their attitude towards transportation services; a function they now view as a key element of their business strategy. The advent of new mobile technologies and online platforms, the use of intermodal logistics, and the multiplication of customer-selected delivery options continue to prompt the development of large-scale complex transportation models. The scope of such models can address a single tier of the supply chain or lie at the interface of two tiers when this integration is necessary to reveal important managerial tradeoffs. Such problems require cutting-edge optimization techniques and powerful computing platforms. Given the scale and recurrence of logistical operations, data-driven optimized policies can achieve multi-million dollar savings in cost and significant improvement in service level. This dissertation develops, in its three essays, specialized algorithms for solving two integrated routing problems that have applications in bi-level transportation.

Essay One proposes an exact branch-cut-and-price algorithm for the generalized vehicle routing problem (GVRP) which has applications in maritime transportation, survivable telecommunication network design, and health-care logistics. Decomposition techniques are used to reformulate the GVRP as a set-partitioning model which prompts the development of a column generation approach. A specialized dynamic programming algorithm is proposed for solving the pricing sub-problem. The performance of the proposed algorithm is significantly improved by enforcing a set of rounded capacity valid inequalities. Computational results show that the proposed algorithm compares favorably against the state-of-the-art exact algorithm for the GVRP and closes 8 out of 9 previously open GVRP instances in the literature.

Essay Two investigates a variant of the Vehicle Routing-Allocation Problem that arises in the distribution of pallets of goods by a food bank to a network of relatively distant nonprofit organizations. Vehicles are routed to selected intermediate delivery sites to which the nonprofit organizations travel to collect their demand. The logistical cost is shared and the objective is to minimize a weighted average of the food bank vehicle routing cost and the travel cost of the nonprofit organizations. We develop an efficient multi-start heuristic that iteratively constructs initial solutions to this problem and subsequently explores their neighborhoods via local improvement and perturbation schemes. In our experience, the proposed heuristic substantially outperforms alternative optimization-based heuristics in the literature in terms of the solution quality and computational efficiency and consistently yields solutions with an optimality gap of 0.5% on average.

Essay Three develops an effective branch-and-price algorithm for the aforementioned food bank vehicle routing problem. The pricing subproblem is solved, exactly or heuristically, using a specialized labeling type dynamic programming (DP) algorithm. The computational efficacy of this DP approach stems primarily from the inclusion of preprocessing routines that enhance the label extension scheme by iteratively eliminating dominated (partial) solutions. The proposed exact DP algorithm, and five proposed heuristic variants, significantly reduce the computational time associated with the solution of the pricing subproblem (as opposed to solving the latter as a mixed-integer model with CPLEX). The resulting speedup enables the implementation of a branch-and-price algorithm that greatly outperforms the use of CPLEX over a test-bed of 60 problem instances.

TABLE OF CONTENTS

Page

ABSTRACT . . . . **iv**

LIST OF TABLES . . . . **x**

LIST OF FIGURES . . . . **xii**

CHAPTER

1. INTRODUCTION AND MOTIVATION **1**

1.1 Logistics and Supply Chain 2
1.2 Importance of Transportation 2
1.3 Literature Review 3

1.3.1 Generalized Vehicle Routing Problem 6
1.3.2 Vehicle Routing with Demand Allocation Problems 7

1.4 Organization of Dissertation 8

2. A BRANCH-AND-CUT-AND-PRICE ALGORITHM FOR THE GENERALIZED VEHICLE ROUTING PROBLEM **10**

2.1 Introduction and Motivation 10
2.2 Branch-and-Cut-and-Price Algorithm 12

2.2.1 Set Partitioning Formulation 12
2.2.2 Valid Inequalities for Master Program 13
2.2.3 Branching Strategy and Primal Bounds 15

2.3 Solving the Pricing Subproblem 16

2.3.1 Preliminaries 16
2.3.2 An Exact Dynamic Programming Algorithm 20
2.3.3 Heuristic Dynamic Programming Schemes 25

2.4 Computational Study 26

2.4.1 Problem Instances 26
2.4.2 Results for Benchmark Instances 27
2.4.3 Results for Instances with Random Clustering 30

2.5 Conclusion 34

3. A MULTI-START OPTIMIZATION-BASED HEURISTIC FOR A FOOD BANK DISTRIBUTION PROBLEM 35

3.1 Introduction and Motivation 35
3.2 Multi-Start Heuristic 38

3.2.1 Notation and Problem Statement 38
3.2.2 Overall Algorithm 39
3.2.3 Initialization Procedure 40

3.2.3.1 Customer Assignment 41
3.2.3.2 Vehicle Routing 42
3.2.3.3 Solution Recovery 42

3.3 Local Improvement & Perturbation Schemes 43

3.3.1 Procedure Start 43
3.3.2 CVRP Heuristic 44
3.3.3 Procedure GAP: Optimal Assignment of Customers to Routes 45
3.3.4 Delivery Site Removal 46
3.3.5 Solution Perturbation 47

3.4 Computational Study 48

3.4.1 Description of Testbed & Algorithmic Settings 48
3.4.2 Discussion of Results 50

3.4.2.1 Baseline Heuristic Performance 50
3.4.2.2 Sensitivity with Respect to Objective Weights 53
3.4.2.3 Comparative Analysis of Different Algorithmic Schemes 59

3.5 Conclusion 60

4. A BRANCH-AND-PRICE ALGORITHM FOR A VEHICLE ROUTING-ALLOCATION PROBLEM 63

4.1 Introduction and Motivation 63
4.2 Branch-and-Price Algorithm 66

4.2.1 Problem Statement and Set Partitioning Formulation 68
4.2.2 Initial Solution 69
4.2.3 Pricing Subproblem 70
4.2.4 Branching Strategy 72

4.3 Dynamic Programming Algorithm for the Pricing Subproblem 72

4.3.1 Overall Exact DP Approach 73
4.3.2 Path Extension Scheme 80

4.3.2.1 Creating Feasible Customer Subsets 82
4.3.2.2 Dominance Rule 84

4.3.3 Heuristic DP Procedures 84

4.4 Computational results 86

4.4.1 Description of Problem Instances 87
4.4.2 Performance of B&P Algorithm vs. CPLEX 88

4.4.2.1 Usefulness of DP Variants 90

4.4.3 Comparison of Root-Node B&P Solution with Decomposition Heuristics 91

4.5 Conclusions 94

5. CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH 96

5.1 Summary of Findings 96
5.2 Directions for Future Research 98

APPENDICES

A. MIP FORMULATION FOR THE VRDAP 99
B. NOTATIONS 102
C. NUMERICAL EXAMPLE FOR THE DP ALGORITHM INTRODUCED IN CHAPTER 4 104

BIBLIOGRAPHY 108

LIST OF TABLES

Table **Page**

2.1 Computational results for $\theta = 2$ 31

2.2 Computational results for $\theta = 3$ 32

2.3 Effect of random clustering on difficulty of instances for $\theta = 2$ 33

2.4 Effect of random clustering on difficulty of instances for $\theta = 3$ 33

3.1 Parameter values in the proposed heuristic 50

3.2 Heuristic performance for Set A, $\lambda = 0.5$ 51

3.3 Heuristic performance for Set B, $\lambda = 0.5$ 52

3.4 Heuristic performance for Set A, $\lambda = 0.75$ 54

3.5 Heuristic performance for Set B, $\lambda = 0.75$ 55

3.6 Heuristic performance for Set A, $\lambda = 0.25$ 56

3.7 Heuristic performance for Set B, $\lambda = 0.25$ 57

3.8 Comparative Analysis of Different Algorithmic Schemes 61

4.1 Original customer assignment cost matrix (f) 73

4.2 Original routing cost matrix (c) 73

4.3 Dual values associated with constraints (4.1b) 76

4.4 Dual values associated with constraints (4.1c); $\bar{\pi}_0$ is the dual value of (4.1d) 76

4.5 Assignment cost matrix after applying dual values 76

4.6 Routing cost matrix after applying dual values 76

4.7 Comparative results for CPLEX vs. branch-and-price algorithm over 60 instances 92

4.8 Comparative results for the heuristic and exact DP algorithms 93

LIST OF FIGURES

Figure **Page**

1.1 General multi-vehicle VRAP 4

2.1 For cutset $S = \{1, 2\}$, $\gamma_{r_1}(S) = 2$ and $\gamma_{r_2}(S) = 2$ 15

2.2 An illustrative example 17

2.3 Cost matrix 18

2.4 A GVRP instance 18

2.5 The GVRP instance after $CM(0, A)$ 18

2.6 The GVRP instance after $CM(A, B)$ 19

2.7 The GVRP instance after $CM(r)$ 19

4.1 Network structure of the VRAP variant under investigation 64

4.2 Solid lines represent routing and assignment decisions made thus far in subproblem 75

CHAPTER 1

INTRODUCTION AND MOTIVATION

Logistics is broadly defined as the activities required for the movement and handling of goods and materials, from inputs through production to consumers and waste disposal. The performance of the logistics system has a major impact on cost structures, revenues and service quality. Transportation is the most important component of logistics system. In fact, transportation comprises one third to two third of logistics costs. In the United States, companies spend more than $800 billion each year on transportation. It is therefore no surprise when a 2014 Bloomberg survey reports that 73% of supply chain managers are undergoing a shift in their attitude toward transportation and recognizing transportation as a real point of competitive differentiation. Considering the high transportation costs, a very small improvement in transportation operations can sometimes lead to millions of dollars costs saving for organizations. To achieve such improvements, transportation must be optimized and this is achieved by means of mathematical programming algorithms and optimization techniques. In this chapter we first in section 1.1 briefly review the history of logistics. Section 1.2 highlights the importance of transportation. Section 1.3, provides a brief literature review on integrated routing problems; the class of problems that are closely related to the applications we study in this research. Section 1.4 summarizes the organization of the dissertation.

1.1. Logistics and Supply Chain

Logistics was initially used in handling military activities. In military science, logistics is concerned with maintaining army supply lines while disrupting those of the enemy. The birth of military logistics can be traced back to the war times of Greek and Roman empires, where there were officers titled "Logistikas" with the responsibility of providing services related to supply and distribution of resources. Such services were crucial and could simply change the outcome of war because an armed force without resources and transportation is defenseless. Military logistics was greatly improved during the World War II and advanced implementations of logistics were developed.

Despite the long history of military logistics, it took a long time for organizations to recognize the crucial role of logistics in business. In 1962 Drucker described the logistics as "the most sadly neglected, most promising area of businesses". In 1970s logistics started gaining more and more attention among researchers and organizations. This was mostly due to the rise in petroleum price in 1973 which made logistics a very important segment of business accounting for 15-20 percent of costs of organizations.

1.2. Importance of Transportation

Transportation system is the most important economic activity among the components of business logistics systems which comprises around one third to two thirds of the expenses of enterprises logistics costs. In order for organizations to fully benefit from advantages of logistics, a well-developed transportation system is necessary. Nowadays, transportation is considered by organizations to be a competitive differentiator that can help companies to outpace their competition. As such, demand for advanced transportation optimization systems for reaching a cost efficient transportation system in on the rise. Transportation optimization can be separated into two categories, namely, strategic and tactical transportation optimization. At the strategic level, companies make strategic decisions like "desired service level", "addition of new products", "outsourcing the transportation", "budgeting needs", etc. Tactical transportation optimization, is a process that runs daily and make use of specialized mathematical programming algorithms to find the most cost efficient ways for running the transportation related operations, given a set of constraints, e.g. "costs", "equipment availability", "desired service level", etc. This study focuses on transportation optimization at tactical level and investigates two integrated routing problems that have several applications in bi-level transportation. In the next section, we briefly review the literature of problems closely related to our study.

1.3. Literature Review

There is a rich body of literature concerning transportation optimization, from classic Traveling salesman Problem to modern routing applications like Electric Vehicle routing problem or Vehicle Routing with Drones. The problems discussed in the present study fall under the umbrella of the vehicle routing-allocation problem (VRAP). Vehicle routing-allocation problems (VRAP) are concerned with the effective delivery of goods and services and arise in a broad spectrum of applications, including the planning of bus stops (Schittekat et al. 2013), the location of post boxes (Labbé and Laporte 1986), and the training of military units (Nagy and Salhi 2007). As discussed in Beasley and Nascimento (1996), the VRAP is concerned with vehicle routing problems (VRP) where a subset of customers may be directly visited (as in a classical VRP), whereas some other customers may not be visited at all or may be allocated to another customer location that is included in a tour. In the latter cases, a penalty is incurred for not serving a customer or for requiring them to travel to a neighboring location to receive service. Figure 1.1 illustrates a general multi-vehicle VRAP where six customers are directly included in the tours, four are

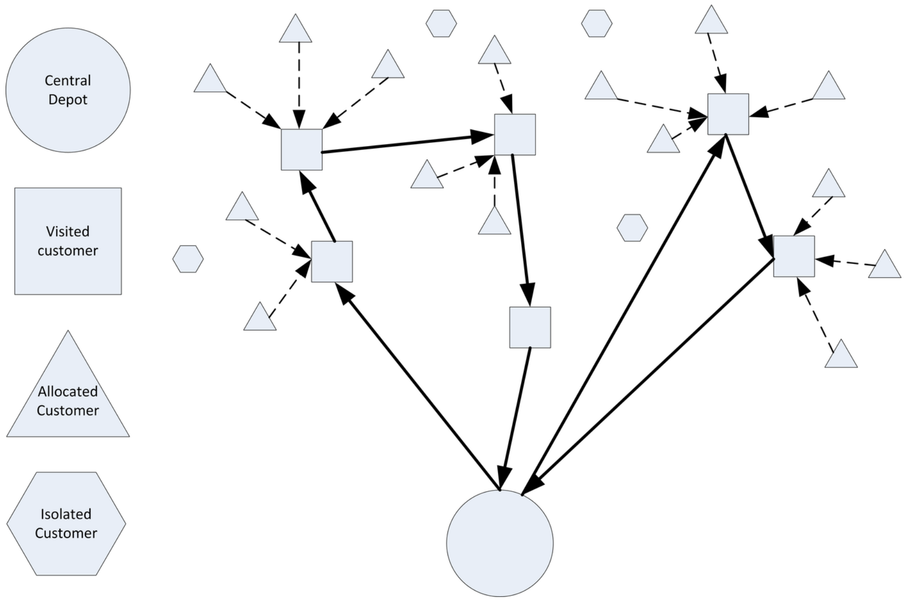

Figure 1.1: General multi-vehicle VRAP

isolated (not served), and the remainder of the customers have to travel to a neighboring customer location that is included in a tour. In the following, we briefly review the literature of the VRAP and highlight the modelling commonalities between the VRAP in general, the specific VRAP variants we examine, and other relevant models, such as the median cycle problem or the capacitated m-ring-star problem. Sections 1.3.1 and 1.3.2 discuss two VRAP variants examined in this study.

Beasley and Nascimento (1996) focus on the single-vehicle VRAP (SVRAP) as defined above. The authors discuss how the SVRAP subsumes and generalizes many problems in the literature that are discussed under various assumptions, including the classical traveling salesman problem (TSP), the covering tour problem, the covering salesman problem (Gendreau et al. 1997), and a variant of the prize collecting TSP. In an early study, Labbé and Laporte (1986) consider an SVRAP setting for the location of post boxes, which they refer to as a *location-allocation-routing problem*. In this context, a subset of post box locations is selected and included in vehicle tours and customers are assigned to nearby post boxes. A mixed-integer program

(MIP) is formulated with the objective of balancing the post collection effort (routing cost) and customer convenience (customer allocation cost) and is tackled using a heuristic. Likewise, Akinc and Srikanth (1992) investigate a similar SVRAP in the context of routing a mobile service facility. The authors develop a Lagrangian-based branch-and-bound algorithm that produces encouraging results for randomly generated problem instances with up to 100 customers. Considering the SVRAP in Beasley and Nascimento (1996), Vogt et al. (2007) propose a tabu search which is compared against other heuristic techniques in the literature.

A body of works has emerged over the last decade on the so-called *median cycle problem* (MCP), also known as the *ring-star problem* (RSP). The MCP is a variant of the SVRAP where customers cannot be left out without service; they are either directly visited or assigned to a neighboring customer location that lies on a tour. The total routing cost plus assignment cost is often minimized and, in certain cases, the routing cost is minimized, while enforcing an upper bound on the total assignment cost (Peréz et al. 2003, Renaud et al. 2004). In particular, Renaud et al. (2004) devise two heuristics for the MCP – a multistart greedy heuristic and an evolutionary algorithm. Further, Labbé et al. (2005) develop an MIP formulation for which classes of valid inequalities are derived and embedded within a branch-and-cut algorithm. The proposed methodology is tested using TSP Library instances and a case study related to the city of Milan. The MCP is further generalized in a multi-vehicle variant, the capacitated m-ring-star problem (CmRSP) (e.g., Baldacci et al. 2007, Naji-Azimi et al. 2010, Hoshino and de Souza 2012).

Another related problem is the covering salesman problem (CSP). In CSP, each customer can cover a subset of customers that are within its pre specified covering radius. The goal is to construct a minimum cost Hamiltonian cycle over a subset of vertices such that all the customers are either located on the constructed cycle or are covered by at least one of the visited customers. Current and Schilling (1989) first introduced CSP and referred to its application in rural healthcare delivery problem. State of the art heuristic algorithm for CSP is ant colony optimization algorithm proposed by Salari et al. (2015). Central to the efficacy of this algorithm is a dynamic programming based heuristic that simultaneously revises the decisions about visited customers and their sequence.

Covering tour problem (CTP) is a generalization of CSP where the set of vertices is divided into three groups, namely, the set of vertices that must be visited, the set of vertices that must be covered and the set of vertices that must be either visited or covered. Gendreau et al. (1997) proposed a heuristic and an exact branch-and-cut algorithm for CTP.

1.3.1 Generalized Vehicle Routing Problem

The Generalized Vehicle Routing Problem (GVRP) deals with delivering goods to a single location (e.g., ports) from every cluster of customers. The underlying assumption is that a second-tier delivery would subsequently take place from these designated locations to every other customer in their respective clusters (Reihaneh and Ghoniem 2018).

The GVRP has several applications in bi-level distribution problems. For example, it arises in maritime transportation (Bektas et. al. 2011) when ports are clustered into several demand regions and vessels are routed to supply exactly one port in every region. In such applications, it is assumed that goods supplied to a selected port will subsequently be distributed during a second phase to other ports in the region, though the second-tier distribution operations are not directly accounted for in the GVRP itself. Bektas et al. (2011) also discussed applications in health-care logistics, urban waste collection and the design of survivable telecommunication networks. Baldacci et al. (2010) also discussed how such problems as the TSP with profits, the VRP with selective backhauls, and the covering VRP, among others, can be modeled as a GVRP.

The Generalized Traveling Salesman Problem (GTSP) is a special case of the GVRP with only one vehicle. The state-of-the-art exact solution approach for the GTSP is a branch-and-cut (BC) algorithm by Fischetti et al. (1997). Reihaneh and Karapetyan (2012) and Karapetyan and Gutin (2012) proposed heuristic approaches for the GTSP and overviewed the extant literature for this problem. While there exists a rich body of works on the GTSP, the GVRP has received limited attention in the literature. Ghiani and Improta (2000) proposed a transformation of the GVRP into an arc-routing problem and demonstrated their methodology using one illustrative example. The state-of the-art exact algorithm for the GVRP is a BC algorithm by Bektas et al. (2011). In addition to the well-known capacity cuts and valid inequalities automatically generated by CPLEX (e.g., Gomory cuts, disjunctive cuts, etc.), the authors also proposed so-called "same vertex" inequalities with the objective of prohibiting the formation of disconnective solutions (a solution that visits more than one node from a cluster).

1.3.2 Vehicle Routing with Demand Allocation Problems

Motivated by food bank operations, Ghoniem et al. (2013) and Solak et al. (2012) consider a *single-depot*, multi-vehicle VRAP with the requirement that all customers be allocated to a subset of selected drop sites. In Ghoniem et al. (2013), a relax-and-fix heuristic (Wolsey 1998) with symmetry-defeating constraints (Sherali and Smith 2001) and a column generation (CG) approach that is accelerated using the complementary column generation feature by Ghoniem and Sherali (2009). Over problem instances with up to 10 candidate drop sites and 50 customers, the CG approach produced better results with optimality gaps of about 4%. However, the computational effort required to solve the pricing subproblem using CPLEX became relatively intense, with an accompanying long CG tailing-off effect. Solak et al. (2012) compared the performance of CPLEX against three heuristics within a time limit of one CPU hour for instances having up to 25 candidate sites and 50 customers. CPLEX produced solutions having an optimality gap of 12.8% at an average, whereas a sequential heuristic worsened the results by CPLEX by nearly 4%. In contrast, a logic-based Benders decomposition approach (Hooker and Ottoson 2003) slightly improved the results of CPLEX by 0.6% and a classical Benders decomposition achieved a 3.6% reduction in the objective value at an average. One major disadvantge, however, is that solutions still exhibited a relatively large optimality gap after a computational time of one CPU hour. Similarly, the planning of school bus stops and routes constitutes another application for the single-depot, multi-vehicle VRDAP where the decision-maker specifies bus stops, assigns students (or, more generally, passengers) to convenient stops, and determines bus routes. Schittekat et al. (2013) propose an MIP model and a metaheuristic approach that yields good quality solutions to instances with up to 800 students and 80 bus stops.

Murty and Djang (1999) tackle a challenging *multi-depot*, multi-vehicle VRDAP variant that arises in scheduling training programs for National Guard Units (NGUs). The problem aims at locating home bases for mobile trainers, secondary locations to which the trainers will travel to provide training, and the allocation of NGUs to nearby home bases or secondary sites for training purposes. The objective is to minimize the routing cost for mobile trainers and the travel cost for the NGUs. Although the authors employ a sequential heuristic, their solution is reported to achieve a 70% savings in total mileage over a benchmark solution developed by the army.

1.4. Organization of Dissertation

This dissertation is organized as follows. In Chapter 2, we develop a branch-and-cut-and-price algorithm for the generalized vehicle routing problem. A specialized dynamic programming algorithm is proposed for solving the pricing sub-problem. Also, the performance of the proposed algorithm is significantly improved by introducing set of valid inequalities and also by designing a heuristic algorithm which seeks near optimal primal bounds for the problem. Computational results show that the proposed algorithm compares very well against the state of the art exact algorithm for GVRP and closes 8 out of 9 previously open GVRP instances in the literature.

Chapter 3 studies vehicle routing with demand allocation (VRDAP), a variation of VRP that has application in food bank distribution planning. A dynamic programming-based branch-and-price (BP) algorithm is developed for VRDAP. The proposed algorithm is demonstrated to greatly outperform the use of commercial branch-and-bound/cut solvers such as CPLEX. The methodology and computational results in this study also substantially improve on recent works in the context of food bank operations where this VRAP is tackled via optimization-based heuristics. Central to the efficacy of the proposed BP algorithm is the development of a specialized dynamic programming procedure that extends works on elementary shortest problems with resource constraints in order to solve the more complex single-vehicle VRAP that underlies the pricing subproblem.

Chapter 4 develops a branch-and-price algorithm for the aforementioned food bank routing problem. Using Dantzig-Wolf decomposition the problem is reformulated as a set-partitioning model and is solved using the column generation technique. The pricing subproblem is solved using a labeling type dynamic programming (DP) algorithm. The computational efficiency of the proposed algorithm is significantly improved by developing five heuristic variants of the DP algorithm. The resulting speedup enables the implementation of a branch-and-price algorithm that greatly outperforms the use of CPLEX over a test-bed of 60 problem instances.

Chapter 5 concludes the dissertation by summarizing our findings and discusses directions for future research.

CHAPTER 2

A BRANCH-AND-CUT-AND-PRICE ALGORITHM FOR THE GENERALIZED VEHICLE ROUTING PROBLEM

This chapter investigates the Generalized Vehicle Routing Problem (GVRP) – an integrated routing problem in which customers are partitioned into mutually exclusive clusters and demand of each cluster of can be dropped off at any of its customers. In GVRP, the decision-maker seeks to determine minimum cost routes using a limited number of vehicles such that every cluster is visited by exactly one route and within any cluster a single customer is visited, subject to vehicle capacity constraints. We develop a branch-and-cut-and-price algorithm for the GVRP. The pricing subproblem is solved using a specialized dynamic programming algorithm. Computational results show that the proposed algorithm compares favorably against a state-of-the-art branch-and-cut algorithm and solves to optimality eight previously open GVRP instances in the literature.

2.1. Introduction and Motivation

The Generalized Vehicle Routing Problem (GVRP) deals with delivering goods to a single location (e.g., ports) from every cluster of customers. The underlying assumption is that a second-tier delivery would subsequently take place from these designated locations to every other customer in their respective clusters (Reihaneh and Ghoniem 2018). The GVRP involves a central depot, denoted by 0, and a set of customers $N = \{1, \dots, n\}$ that are partitioned into m mutually exclusive clusters $H = \{C_1 \dots, C_m\}$ (i.e., $\bigcup_{k=1}^{m} C_k = N$ and $C_{k_1} \cap C_{k_2} = \emptyset$, $\forall k_1 \neq k_2 \in \{1, \dots, m\}$).

Each cluster C has a demand, denoted by q_c, that is fullfilled by one of w identical vehicles having a capacity Q. A feasible route starts at the depot, visits exactly one customer from a subset of clusters such that the aggregate demand of these clusters does not violate the vehicle capacity, and returns to the depot. Denote by $E = \{\{i, j\} | i, j \in N \cup \{0\}, i < j\}$ the set of edges in the network and let $d_{i,j}$ be the cost associated with edge $\{i, j\} \in E$. The decision-maker seeks a minimum cost solution such that exactly one customer in any cluster is visited by exactly one vehicle, no more than w routes are constructed, and vehicle capacity constraints are satisfied.

This chapter proposes a branch-and-cut-and-price (BCP) algorithm for the GVRP that compares favorably against the state-of-the-art BC algorithm (Bektas et al. 2011). The solution scheme is driven by the addition of rounded capacity cuts to the master program and the development of a specially-tailored dynamic programming (DP) approach to solve the column generation pricing subproblem. Although our computational study indicates that no algorithm systematically dominates the other over classical benchmark instances, the proposed BCP enabled optimal solutions to eight benchmark instances that were previously open in the literature.

The remainder of this chapter is organized as follows. In section 2.2, the key elements of the BCP algorithm are presented. First, the problem is formulated as a set partitioning formulation to which cuts are appended. Second, the branching mechanism and the heuristic employed for obtaining primal bounds are briefly discussed. In Section 2.3, the DP approach for solving the pricing subproblem is presented in detail. Our computational study in Section 2.4 compares our BCP algorithm against the BC algorithm using classical benchmark instances and instances constructed using a random clustering scheme. Section 2.5 concludes the chapter with a summary of our findings.

2.2. Branch-and-Cut-and-Price Algorithm

The GVRP is modeled as a set partitioning formulation in Section 2.2.1. Rounded capacity inequalities that can strengthen the underlying set partitioning formulation are discussed in Section 2.2.2. Section 2.2.3 presents our branching scheme and a heuristic approach that is employed in order to generate upper (primal) bounds.

2.2.1 Set Partitioning Formulation

Let $\Re$ be the set of all feasible routes for a GVRP instance and $\Re_c$ be the set of all routes that include cluster C. For any route $r \in \Re$ and cluster C, we introduce the coefficient α_c^r which takes a value of 1 if r visits C and $\alpha_c^r = 0$ otherwise. Let ρ_r be the cost associated with route $r \in \Re$. The GVRP can be modeled as the following set partitioning formulation, denoted by **SPP**, where the binary variable z_r indicates whether route r is selected in the solution or not:

$$\textbf{SPP:} \text{ Minimize} \sum_{r \in \Re} \rho_r z_r \tag{2.1a}$$

$$\text{subject to} \sum_{r \in \Re} \alpha_c^r z_r = 1 \qquad \forall\, C \in \{C_1, \ldots, C_m\} \tag{2.1b}$$

$$\sum_{r \in \Re} z_r \leq w \tag{2.1c}$$

$$z \text{ binary.} \tag{2.1d}$$

The objective function (2.1a) minimizes the routing cost. Constraint (2.1b) ensures that every cluster is served by exactly one vehicle route, thereby achieving a partitioning scheme for clusters. Constraint (2.1c) enforces an upper bound on the number of routes (vehicles). While each column in SPP only represents the set of clusters visited by a vehicle tour, the specific customer visited in each of these clusters and the sequence in which they are visited is captured in the cost of the column.

Because Model SPP typically involves an exponential number of feasible routes, it is not practical to generate the entire set $\Re$ and column generation (CG) approaches offer a judicious alternative. In this context, Model SPP serves as a master program (MP). The linear programming (LP) relaxation of the MP is solved by initializing a restricted master program (RMP) with a limited set of columns, $\hat{\Re}$, and by iteratively solving a pricing subproblem in order to identify promising columns (having a negative reduced cost) to be added to $\hat{\Re}$. This process continues until no additional worthwhile columns can be constructed by solving the pricing subproblem. The resulting LP solution of the RMP yields a lower bound (LB) for the optimal objective value of the MP. If the LP solution turns out to be binary-valued, it is indeed optimal. Otherwise, rounds of valid inequalities are iteratively added to the RMP in order to strengthen the obtained lower bound, as discussed next in Section 2.2.2. If no additional cuts are identified for inclusion in the RMP formulation and the LP solution of the RMP is still fractional, branching takes place as discussed in Section 2.2.3.

2.2.2 Valid Inequalities for Master Program

We enforce rounded capacity inequalities in the proposed BCP algorithm in order to strengthen the LB associated with the RMP. Capacity inequalities for the GVRP impose a minimum number of vehicles that are needed for any subset of clusters. Recall that $H = \{C_1, \ldots, C_m\}$ and let $H' = \{C_0, C_1, \ldots, C_m\}$ where $C_0 = \{0\}$ is a fictitious cluster containing the depot only. For any C_i and C_j in H', $i < j$, we define super-edge $E_{\{C_i, C_j\}}$ to capture the existence of at least one edge between C_i and C_j. Let $\Psi = \{S \subseteq H : |S| \geq 2\}$ and $\delta(S)$, for any $S \in \Psi$, be the set of all super-edges that have one end in S and the other in $H' \setminus S$. A lower bound on the number of vehicles required to serve $S \in \Psi$ is $k(S) = \lceil q(S)/Q \rceil$, where $q(S)$ is the total demand of all clusters in S. The flow along any edge $e \in E$ is represented by a binary variable y_e which equals 1 if and only if edge e is visited in the GVRP solution. For any super-edge $E_{\{C_i,C_j\}}$, we define x_{ij} to be the summation of the flows over all the edges between C_i and C_j. In other words, x_{ij} is summation of flow of all the edges that have one end in C_i and the other end in C_j. The rounded capacity inequality can be stated as follows:

$$\sum_{\{C_i,C_j\}\in\delta(S)} x_{i,j} \geq 2k(S), \quad \forall S \in \Psi. \tag{2.2}$$

This inequality can be reformulated in the context of Model SPP as follows:

$$\sum_{r\in\Re(S)} \gamma_r(S) z_r \geq 2k(S), \quad \forall S \in \Psi, \tag{2.3}$$

where $\Re(S)$ is the set of all the routes in $\Re$ that visit at least one cluster in S. Moreover, the parameter $\gamma_r(S) = \sum_{\{C_i,C_j\}\in\delta(S)} \eta^r_{i,j}$ where $\eta^r_{i,j}$ is defined as follows:

- If r visits only the depot (i.e. C_0) and a single cluster $C_j \in H$, $\eta^r_{0,j} = 2$ and $\eta^r_{0,h} = 0$, $\forall C_h \in H \setminus \{C_j\}$.
- If r visits C_0 and at least two other clusters, $\eta^r_{i,j} = 1$ if C_i and C_j immediately follow each other in r and otherwise $\eta^r_{ij} = 0$.

To illustrate, assume that $S = \{C_1, C_2\}$ and consider routes r_1 and r_2 as depicted in Figure 2.1. Because only the sequence of clusters in a route is relevant here, we have not explicitly depicted customers within each clusters. Route r_1 only visits cluster C_1 and therefore, $\eta^{r_1}_{0,1} = 2$. Since $E_{\{C_0,C_1\}} \in \delta(S)$, $\gamma_{r_1}(S) = 2$. In route r_2, $\eta^{r_2}_{0,1} = \eta^{r_2}_{1,2} = \eta^{r_2}_{2,3} = \eta^{r_2}_{0,3} = 1$. Also, $E_{\{C_0,C_1\}}$ and $E_{\{C_2,C_3\}}$ belong to $\delta(S)$. Therefore, $\gamma_{r_2}(S) = 2$.

To circumvent the generation of an exponential number of rounded capacity inequalities, a separation routine examines the LP relaxation solution of the RMP in order to identify violated capacity inequalities in an iterative fashion. We used in our implementation the separation routines made available in the CVRPSEP package by

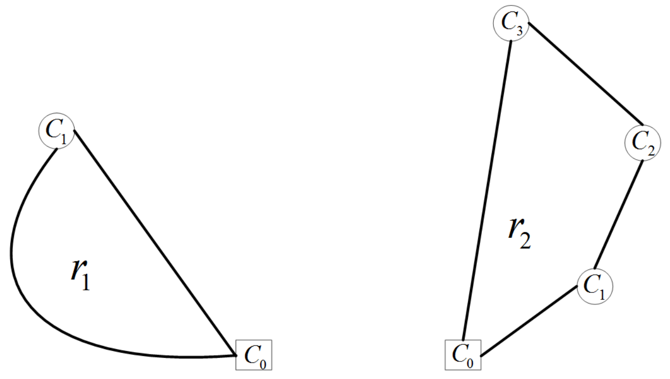

Figure 2.1: For cutset $S = \{1,2\}$, $\gamma_{r_1}(S) = 2$ and $\gamma_{r_2}(S) = 2$

Lysgaard et al. (2004). We also considered enforcing weak subset-row inequalities as introduced by Baldacci et al. (2011). However, our preliminary computational results suggest that these were not found to improve the quality of the lower bounds obtained by our BCP algorithm for the tested benchmark instances and are, therefore, not included in our presentation or final results. The mere inclusion of capacity inequalities yielded relatively tight lower bounds at the root of the BCP algorithm, resulting in an optimal solution at the root node itself for many benchmark instances.

2.2.3 Branching Strategy and Primal Bounds

The proposed BCP algorithm employs a best first branching strategy using the super-edge flow variables, i.e., the x_{ij} variables. First, the flow along any super-edge is computed from the LP solution of the RMP at hand. The super-edge $E_{\{C_i,C_j\}}$ having a most fractional flow, that is, a flow value closest to 0.5, is selected for branching. Two new child nodes are generated where in one child node, clusters C_i and C_j should be visited immediately after each other. In the other child node, clusters C_i and C_j cannot be visited immediately after each other in any route.

Upper bounds are calculated by adapting the heuristic of De Franceschi et al. (2006). A GTSP local search called *Cluster Optimization(CO)* (Renaud and Boctor 1998) is also used to enhance the heuristic solution. Furthermore, all the routes in up to 20 best solutions found by the heuristic are used in order to initialize the set of columns in the RMP.

2.3. Solving the Pricing Subproblem

The pricing subproblem of GVRP is NP-Hard because it generalizes the pricing subproblem of the capacitated VRP (CVRP) which is known to be NP-hard (Desrochers et al. 1992). We first derive some preliminary observations and results in Section 2.3.1 which serve as a cornerstone for the DP algorithm that we develop in Section 2.3.2 for solving the pricing subproblem.

2.3.1 Preliminaries

This section introduces some preliminary results that enable the computation of a shortest cost associated with an ordered sequence of clusters to be visited in a route. This concept forms a foundational block for the development of the DP algorithm.

Arc Cost Transformation. For any node $v \in N$, we define a scalar λ_v that affects the cost of all its incident arcs. Specifically, the value λ_v is respectively added to and subtracted from the cost of all arcs leaving and entering v. Accordingly, the modified cost of arc (i, j), denoted by $\bar{d}_{i,j}$, is defined as follows: $\bar{d}_{i,j} = d_{i,j} + \lambda_i - \lambda_j$, where d_{ij} is the original cost of (i, j). Considering the illustrative example in Figure 2.2, customers are represented by small letters (a,. . .,e) and the numbers next to them specify their associated λ values. The original symmetric arc costs are also represented in Figure 2.2. Upon applying the arc cost transformation scheme using these λ values, we obtain the following modified costs: $\bar{d}_{a,c} = 30 + 10 - 5 = 35$, $\bar{d}_{c,a} = 30 + 5 - 10 = 25$, $\bar{d}_{b,e} = \bar{d}_{e,b} = 15$, $\bar{d}_{b,d} = 10 + 0 - 15 = -5$, $\bar{d}_{d,b} = 10 + 15 - 0 = 25$.

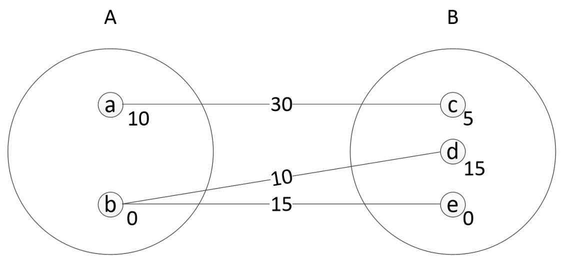

Figure 2.2: An illustrative example

Proposition 1. Applying the arc cost transformation scheme does not change the cost of any feasible solution for the GVRP.

Proof. Any GVRP route r that enters a customer node v must also leave it. Therefore, for any node v in r, its λ_v value gets added to and substracted from the cost of r, resulting in no change in the cost of any GVRP solution. $\square$

Connective Move for a Pair of Clusters. We define the *minimum arc cost* from cluster A to cluster B as $\bar{d}_{A,B} = \min\{\bar{d}_{u,v} | u \in A, v \in B\}$. In Figure 2.2, $\bar{d}_{A,B} = \bar{d}_{b,d} = 10 + 0 - 15 = -5, \bar{d}_{B,A} = \bar{d}_{e,b} = 15$. Similarly, for a cluster A and a node $v \notin A$, we define $\bar{d}_{A,v}$ and $\bar{d}_{v,A}$, where node v is viewed as a singleton cluster $\{v\}$. A *connective move* from cluster A to B, denoted by $CM(A, B)$, considers $\bar{d}_{A,B}$ as the length of the arc between the two clusters and adjusts the λ_v value for any $v \in B$ such that $\bar{d}_{A,v} = \bar{d}_{A,B}$. In other words, arc costs are modified such that the minimum arc cost from cluster A to any customer $v \in B$ equals the minimum arc cost between the two clusters. For instance, consider the GVRP instance represented in Figure 2.4 with an associated cost matrix in Figure 2.3. First, we conduct a connective move from the depot to cluster A. Noting that $\bar{d}_{0,A} = 30$, we obtain $\lambda_a = 35 - 30 = 5$ and $\lambda_b = 30 - 30 = 0$. Applying these λ values will result in the graph in Figure 2.5. Next, we consider the connective move from A to B. Because $\bar{d}_{A,B} = 10$, we

$$\begin{array}{c|cccccccc}
 & 0 & a & b & c & d & e & f & g \\
\hline
0 & - & 35 & 30 & 50 & 30 & 90 & 70 & 30 \\
a & 35 & - & - & 15 & 20 & 45 & 70 & 80 \\
b & 30 & - & - & 20 & 10 & 40 & 60 & 70 \\
c & 50 & 15 & 20 & - & - & - & 25 & 40 \\
d & 30 & 20 & 10 & - & - & - & 30 & 60 \\
e & 90 & 45 & 40 & - & - & - & 10 & 40 \\
f & 70 & 70 & 60 & 25 & 30 & 10 & - & - \\
g & 30 & 80 & 70 & 40 & 60 & 40 & - & -
\end{array}$$

Figure 2.3: Cost matrix

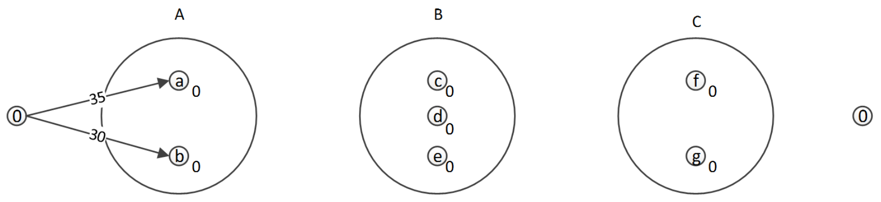

Figure 2.4: A GVRP instance

obtain $\lambda_c = 20 - 10, \lambda_d = 10 - 10$ and $\lambda_e = 40 - 10$. Figure 2.6 shows the graph after applying λ values.

Let $r = (0, C_1, \cdots, C_k, 0)$ be a sequence of clusters in a GVRP route. We define $TSP(r)$ to be the optimal TSP tour that visits the clusters according to r. In other

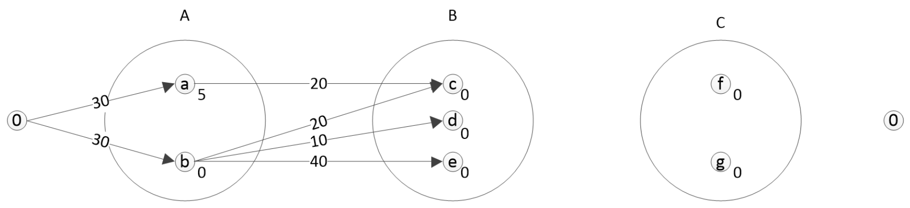

Figure 2.5: The GVRP instance after $CM(0, A)$

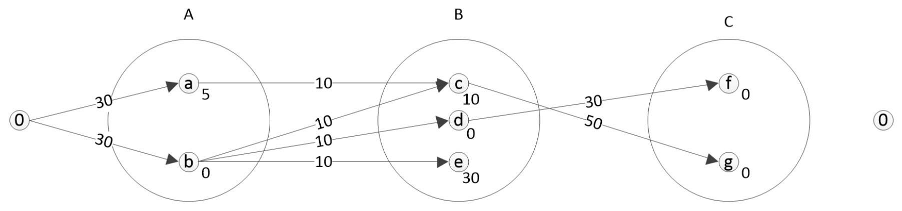

Figure 2.6: The GVRP instance after $CM(A, B)$

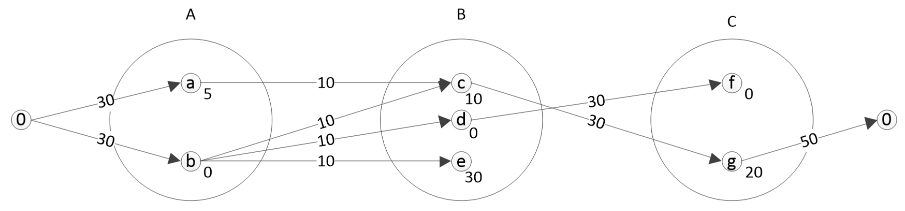

Figure 2.7: The GVRP instance after $CM(r)$

words, $TSP(r)$ is the optimal selection of customers from clusters when visited in the same order as dictated by r. In the previous example, $TSP(r) = (0, a, c, g, 0)$.

Connective Moves for an Ordered Sequence of Clusters. For a sequence starting and ending at node 0 and comprising an ordered sequence of clusters $r = (0, C_1, \cdots, C_k, 0)$, Procedure $CM(r)$ (Algorithm 1) performs connective moves on consecutive elements of this sequence and computes $\bar{d}_{CM(r)} = \bar{d}_{0,C_1} + \sum_{i=1}^{k-1} \bar{d}_{C_i,C_{i+1}} + \bar{d}_{C_k,0}$. Proposition 2 shows that $\bar{d}_{CM(r)}$ is the cost of $TSP(r)$. Considering the GVRP instance in Figure 2.3, Figures 2.4-2.7 display how Procedure $CM(r)$ performs connective moves for $r = (0, A, B, C, 0)$. In this example, $\bar{d}_{CM(r)} = 30 + 10 + 30 + 50$.

Proposition 2. The $\bar{d}_{CM(r)}$ computed in Algorithm 1 is the cost of $TSP(r)$.

Proof. Based on Proposition 1, the arc cost transformation using the λ values in the course of Procedure $CM(r)$ is conducted without loss of optimality. Further, because each step of Procedure $CM(r)$ considers the length of the shortest arc between consecutive clusters, $\bar{d}_{CM(r)}$ is a lower bound on the value of $TSP(r)$. Using the following backtracking technique, we find a feasible TSP tour whose length is

Algorithm 1 PROCEDURE $CM(r)$

```
1: Comment: Let $C_0$ and $C_{k+1}$ both represent depot($C_0 = C_{k+1} = \{0\}$)
2: Input: $r = (0, C_1, \ldots, C_k, 0)$
3: $\lambda_v = 0 \quad \forall v \in N$
4: $\bar{d}_{CM(r)} \leftarrow 0$
5: for $i = 0$ to $k$ do
6:     Compute $\bar{d}_{C_i,C_{i+1}}$, the minimum arc cost from $C_i$ to $C_{i+1}$
7:     $\bar{d}_{CM(r)} \leftarrow \bar{d}_{CM(r)} + \bar{d}_{C_i,C_{i+1}}$
8:     Perform $CM(C_i, C_{i+1})$
9: end for
```

$\bar{d}_{CM(r)}$. We start by letting $T = (0)$. Then, let v_k be a customer in C_k for which $\bar{d}_{v_k,0} = \bar{d}_{C_k,0}$ and update $T = (v_k, 0)$. By design of $CM(r)$, for any customer in C_k (including v_k), there is a $v_{k-1} \in C_{k-1}$ for which $\bar{d}_{v_{k-1},v_k} = \bar{d}_{C_{k-1},C_k}$. We thus update $T = (v_{k-1}, v_k, 0)$. This process is repeated until we obtain the following tour $T = (0, v_1, \cdots, v_k, 0)$ whose cost equals $d_{CM(r)}$ because the shortest arcs have been used between consecutive clusters. Therefore, T is a tour whose length equals the lower bound on $TSP(r)$ and, therefore, constitutes an optimal tour for the inputted sequence of clusters. $\square$

Should ties present themselves, Algorithm 1 breaks them arbitrarily. To illustrate Algorithm 1, consider the GVRP instance in Figure 2.3 and the sequence of ordered clusters $r = (0, A, B, C, 0)$. After performing Procedure $CM(r)$ (as depicted in Figure 2.7) and, starting with $T = (0)$, we note that $\bar{d}_{g,0} = \bar{d}_{C,0}$. The partial tour is therefore updated to $T = (g, 0)$. For cluster B, $\bar{d}_{c,g} = \bar{d}_{B,C}$ and hence, $T = (c, g, 0)$. Following the same logic, node a is selected from cluster A ($T = (a, c, g, 0)$) and finally a complete minimal cost tour $T = (0, a, c, g, 0)$ is obtained.

2.3.2 An Exact Dynamic Programming Algorithm

This section proposes a dynamic programming (DP) algorithm for the CG pricing subproblem using the preliminary results and observations in Section 2.3.1. The objective of the pricing subproblem is to construct routes having a minimum reduced cost. If the minimum reduced cost is negative, the columns associated with such routes (and possibly several other routes having a negative reduced cost) are added to the RMP. Otherwise, the CG procedure terminates with an optimal LP solution for the master program. Let π_C be the dual variable associated with cluster C in Constraint (2.1b), π_{C_0} be the dual variable for the depot cluster $C_0 = \{0\}$ associated with Constraint (2.1c), and β_S be the dual variable associated with the capacity cut (2.3) defined for set S. In order to embed the dual values into arc costs, we update the original cost matrix as follows:

$$d_{i,j} \leftarrow d_{i,j} - \frac{1}{2}(\pi_{\sigma(i)} + \pi_{\sigma(j)}) - \sum_{S|E_{\sigma(i),\sigma(j)} \in \delta(S)} \beta_S \qquad \forall \{i,j\} \in E \tag{2.4}$$

where $\sigma(i)$ is the cluster to which customer i belongs and $\sigma(0) = C_0$ for the depot.

In the proposed DP algorithm, each path is represented by a label denoted by $L = \{P_\ell, Z_\ell, Q_\ell, \Lambda_\ell, \Pi_\ell\}$ where P_ℓ is an ordered set representing the partial path associated with L that starts from the depot and sequentially visits a subset of clusters; Z_ℓ is the cost associated with L; Q_ℓ is the total demand of all clusters visited by P_ℓ; Λ_ℓ stores the λ values of the customers in the last cluster visited by P_ℓ; and Π_ℓ is set of unreachable clusters, i.e., all the clusters that L cannot be extended to. Each cluster is associated with all labels that end at this cluster (i.e., their P_ℓ sets end with this cluster). Each time a cluster is treated all its labels that were not treated yet are extended to reachable clusters.

The proposed algorithm starts at the depot with the initial label $L = \{0, 0, 0, \emptyset\}$. This label is then extended to all clusters, creating a new label (partial path) for each one of them. The newly generated labels are then iteratively extended to other reachable clusters, constructing more partial paths, until no label is available for extension. Let $L = \{P_\ell, Z_\ell, Q_\ell, \Lambda_\ell, \Pi_\ell\}$ be a label with $P_\ell = (0, C_1, \ldots, C_k)$. When extending

L to a cluster X, we first verify whether such an extension is feasible or not. An extension is deemed feasible if the new label does not violate the vehicle capacity constraint, i.e., $Q_\ell + q_X \leq Q$, and cluster X is not currently unreachable for L, i.e., $X \notin \Pi_\ell$. If this extension is indeed feasible, a new label $T = \{P_t, Z_t, Q_t, \Lambda_t, \Pi_t\}$ is created by appending X to the partial path. That is, we let $P_t = (0, C_1, \ldots, C_k, X)$ and $Q_t = Q_\ell + d_X$. The connective move $CM(C_k, X)$ is performed in order to set the values in Λ_t, i.e., the λ values associated with the customers in the last cluster of T. The label cost is set to $Z_t = Z_\ell + \bar{d}_{C_k,X}$, where $\bar{d}_{C_k,X}$ is the shortest cost arc between clusters C_k and X computed using the λ values. Note that the cost of label T is computed by performing connective moves on P_t, which is similar to computing $\bar{d}_{CM(r)}$ in Algorithm 1. In fact, $Z_\ell + \bar{d}_{X,0}$ is the cost of shortest TSP tour that starts from the depot, sequentially visits clusters of P_t, and returns to depot. We next establish the employed dominance rule and discuss how the set of unreachable nodes, Π_t, is determined.

Dominance rule. There is an exponential number of labels that can be feasibly generated during the DP algorithm. However, since only one or a few of these labels are optimal, it is crucial to curtail the proliferation of labels by detecting, as early as possible, dominated ones. A label is deemed to be dominated if it cannot yield an optimal solution for the pricing subproblem. To this end, we employ the following dominance rule. Let L^* and $L^{'}$ be two labels both visiting cluster X as their last cluster. If X has only one customer i, then L^* dominates $L^{'}$ if the following conditions are met:

(i) $Q^*_\ell \leq Q^{'}_\ell$

(ii) $\Pi^*_\ell \subseteq \Pi^{'}_\ell$

(iii) $Z^*_\ell + \lambda^*_i \leq Z^{'}_\ell + \lambda^{'}_i$

where, λ^*_i and $\lambda^{'}_i$ are λ values associated with customer i in labels L^* and $L^{'}$, respectively. If X has multiple customers, the above conditions result in deletion of i from X in $L^{'}$. If during the process of checking the dominance rule for $L^{'}$ and other labels, all customers in X in label $L^{'}$ are deleted, $L^{'}$ can be deleted as a dominated label. Otherwise, the deleted customers will not be considered when extending X in $L^{'}$ to a cluster, say Y. Such deletions of customers in X typically result in greater values for $\bar{d}_{X,Y}$ and $Z^{'}$ and, as a consequence, increase the possibility of identifying dominated labels. We next provide a formal proof for the aforementioned dominance rule.

Proposition 3. Let L^* and $L^{'}$ be two labels both ending at cluster X and satisfying Conditions (i) and (ii). Any customer $i \in X$ for which $Z^{'}_\ell - Z^*_\ell \geq \lambda^*_i - \lambda^{'}_i$ can be removed from the last cluster of $L^{'}$.

Proof. Let $P_{\ell'} = (0, C^{'}_1, C^{'}_2 \ldots, X)$ be the partial path associated with $L^{'}$. Also, let $\bar{P} = (Y, \bar{C}_1, \bar{C}_2, \ldots, 0)$ be the partial path that would result in an optimal completion of $P_{\ell'}$, denoted by $\hat{P} = (0, C^{'}_1, \ldots, X, Y, \bar{C}_1, \ldots, 0)$. If an optimal TSP tour associated with $\hat{P}$ does not visit i from X, then i can be removed from X in $L^{'}$. Now assume that this TSP tour visits customer $i \in X$ then customer j in cluster Y. Because Conditions (i) and (ii) hold, $\bar{P}$ can also be appended to the partial path P_{ℓ^*} of label L^*. Before extending P_{ℓ^*} with $\bar{P}$, all customers are removed from clusters X and Y except customers $i \in X$ and $j \in Y$, respectively. Upon extending $\bar{P}$ to L^*, note that $\bar{d}^*_{i,j} = d_{i,j} + \lambda^*_i \leq d_{i,j} + \lambda^{'}_i + Z^{'} - Z^*$, where $\bar{d}^*$ and $\bar{d}^{'}$ are costs computed based on Λ^*_ℓ and $\Lambda^{'}_\ell$, respectively. With $d_{i,j} + \lambda^{'}_i = \bar{d}^{'}_{i,j}$, we conclude that $\bar{d}^*_{i,j} + Z^*_\ell \leq \bar{d}^{'}_{i,j} + Z^{'}_\ell$. In other words, the cost of extending label L^* with cluster Y is smaller than or equal to that for extending $L^{'}$ with Y. Since cluster Y has only customer, j, $\lambda^*_j = \lambda^{'}_j = 0$ and, hence, the cost of appending the remaining clusters in $\bar{P}$ to $L^{'}$ or L^* is the same. Therefore, if customer i were visited in cluster X in $L^{'}$, then an optimal extension of label $L^{'}$ would be dominated. Therefore, customer i can be removed from X in $L^{'}$

without loss of optimality. $\square$

Unreachable Clusters. Let L be a label with an associated partial path $P_\ell = (0, C_1, \ldots, C_\ell)$. The set Π_ℓ of unreachable clusters for L can be defined in a variety of ways. For example, Π_ℓ could include all clusters already visited by P_ℓ. Although this definition of Π_ℓ ensures the construction of optimal elementary shortest paths, it results in a slower convergence of the DP algorithm. Alternatively, one could simply set $\Pi_\ell = \{C_\ell\}$, thereby forbidding only the last cluster in label L. The consequent dominance rule results in a more computationally efficient DP algorithm, but allows the formation of cycles in optimal solutions of the DP algorithm. This, in turn, would result in weaker lower bounds for the set partitioning formulation (Model SPP). However, several techniques have been proposed in order to tighten the lower bound, with an added computational expense in solving the DP algorithm. To this end, the definition of the set Π_ℓ of unreachable nodes ought to be refined. Ideally, Π_ℓ should include the minimum number of clusters that can prevent the formation of cycles. In our implementation, we compute elementary paths by employing the algorithm of Martinelli et al. (2014) which takes advantage of the concept of *state-space-relaxation* of Righini and Salani (2008) and *ng-sets* of Baldacci et al. (2011). The notion of completion bounds is also used to speed up the algorithm (Baldacci et al. 2011, Martinelli et al. 2014). For each cluster C, we let N_C be the set of clusters that can be remembered by C and define Π_ℓ as follows:

$$\Pi_\ell = \left\{ C_i \in P_\ell \setminus \{C_\ell\} : C_i \in \bigcap_{s=i+1}^{\ell} N_{C_s} \right\} \cup \{C_\ell\} \tag{2.5}$$

In order to obtain shortest elementary paths, N_C of clusters are updated iteratively. First, the DP algorithm is solved with $N_C = \varnothing$ for any cluster C. If an optimal solution of the DP algorithm has cycles, then for each cycle denoted by $\mathcal{C} = \{X, \ldots, X\}$, the repeated cluster X is added to N_C for all $C \in \mathcal{C}$. After updating

N_C for all clusters, the DP algorithm is solved iteratively until an optimal elementary shortest path is obtained.

2.3.3 Heuristic Dynamic Programming Schemes

Solving the pricing subproblem using the DP algorithm can be computationally onerous for certain problem instances. The computational burden can be deemed to be unjustified, noting that many of the hard-to-construct columns will not ultimately be included in an optimal LP solution of the RMP. This is specially true at the initial iterations of the CG procedure. The computational efficiency of the BCP algorithm can, therefore, be improved by using a heuristic variant of the DP algorithm to solve the CG pricing subproblem. If the heuristic DP fails to produce a column having a negative reduced cost, one can then resort to using the exact DP algorithm. Such alternations between using heuristic and exact DP algorithm for solving the subproblem has been reported to be computationally beneficial in a variety of applications (e.g., Dell'Amico et al. 2006, Martinelli et al. 2014; Ghoniem et al. 2015).

To accelerate the solution of the pricing subproblem, two heuristic variants of the DP algorithm, denoted by H1 and H2, are proposed. For Heuristic H1, at each iteration of the CG procedure, Condition (ii) of the dominance rule (in Section 2.3.2) is relaxed. For Heuristic H2, instead of extending a label to all other clusters (as in the exact DP), it is only extended to 50% of these clusters, focusing on the nearest ones only. The average cost of all the edges between two clusters is used to measure the nearness of two clusters. If Heuristic H1 fails to construct a column having a negative reduced cost, Heuristic H2 is invoked. If Heuristic H2 fails, the exact DP algorithm is invoked. Up to 40 columns having the most negative reduced costs are added to the RMP.

2.4. Computational Study

In this section, we examine the computational performance of the proposed BCP algorithm and compare it against the BC algorithm by Bektas et al. (2011). The proposed algorithm was implemented in C# under visual studio. All runs were performed on a PC with Intel Core i7-2600 3.40 GHz CPU and 12 GB RAM. Following Bektas et al. (2011), a time limit of 7200 CPU seconds was imposed on all runs.

In our computational study, two types of instances are considered which are respectively referred to as *base benchmark instances* and *instances with randomly generated clusters*, as described in Section 2.4.1. Results for these two types of instances are reported and discussed in Sections 2.4.2 and 2.4.3, respectively.

2.4.1 Problem Instances

Base benchmark instances and instances with random clustering have been constructed as follows:

- **Base Benchmark Instances.** The benchmark instances in Bektas et al. (2011) were constructed from CVRP instances using the clustering technique in Fischetti et al. (1997). Specifically, three sets are considered – Sets A, B, and P – from the CVRP-library, with a number of vertices ranging from 16 to 101. The name of each instance follows the format X-nY-kZ-CΩ-VΦ, where X specifies the set, Y the number of vertices, Z the number of vehicles in the original CVRP instance, Ω the number of clusters, and Φ the number of vehicles in the GVRP instance. For any CVRP instance with n vertices, a GVRP instance was constructed with $m = \lceil n/\theta \rceil$ clusters, where $\theta = 2$ and $\theta = 3$. First, m customers, as distant from one another, are selected as the center of the m clusters. Each remaining customer is then assigned to its "nearest" cluster such that the cost between the customer and the cluster center is minimal. Demand of each cluster C is calculated as $q_c = \frac{1}{m} \sum_{i \in C} \omega_i$, where ω_i is the demand of customer i in the original CVRP instance. Finally, a bin-packing problems is solved to determine the number of vehicles in the GVRP instance.

- **Instances with Random Clustering.** In order to investigate the effect of clustering on the difficulty of instances for their BC algorithm, Bektas et al. (2011) also generated instances where customers are randomly assigned to clusters without proximity or cost considerations. In generating such instances, Bektas et al. (2011) only considered instances with $n \leq 45$ and $\theta = 2$. However, because these instances are not publicly available, we independently generated similar instances following the aforementioned scheme for Sets A, B, and P with $n \leq 45$ and with $\theta = 2$ and $\theta = 3$.

2.4.2 Results for Benchmark Instances

Tables 2.1 and 2.2 compare the results of the proposed BCP algorithm for $\theta = 2$ and $\theta = 3$, respectively, against those reported by Bektas et al. (2011) for their BC algorithm. In both tables, column UB provides the best (possibly optimal) solution found within the pre-specified time limit of 7200 CPU seconds. Column LB-0 reports the lower bound at the root node of both algorithms (after adding cuts). Columns BB and CPU respectively report the number of branch-and-bound nodes explored and the CPU time in seconds.

Because the two algorithms were implemented under different computing platforms, our discussion cannot involve an instance-by-instance comparison of CPU times. Rather, it focuses on the relative efficiency with which either methodology solved to optimality certain subsets of instances. In our discussion, we deem an instance to be challenging for a particular methodology if it requires over 600 CPU seconds.

Results for $\theta = 2$

The BCP algorithm solved to optimality 5 instances that were previously open in the literature. All of these instances were solved within about 370 CPU seconds and one of them in particular, Instance P-n55-k15-C28-V8, was solved at the root node of the BCP algorithm within 2 CPU seconds. It, however, failed to solve to optimality 3 instances that were solved by the BC algorithm. Only one instance (A-n80-k10-C40-V5) remains unsolved using either methodology. The following detailed observations are made for Sets A, B, and P:

- Except for Instance A-n80-k10-C40-V5, the BCP algorithm solved all instances of Set A within 364 CPU seconds. On the other hand, the BC algorithm failed to solve 2 instances in this set (A-n63-k9-C21-V3 and A-n80-k10-C40-V5) and required a substantial computational effort (over 600 CPU seconds) for 4 other instances.

- Both the BC and BCP algorithms solved all instances in set B to optimality within 2 CPU hours. Except for Instance B-n50-k8-C25-V5, the BC algorithm solved all instances of set B within 249 CPU seconds. On the other hand, instances of Set B were relatively more challenging for the BCP algorithm as it solved 5 of them to optimality in over 600 CPU seconds.

- For set P, the results are also mixed. The BC algorithm failed to solve 4 instances, namely, P-n50-k8-C25-V4, P-n55-k15-C28-V8, P-n60-k10-C30-V5, P-n60-k15-C30-V8, which the BCP algorithm could solve to optimality within 373 CPU seconds. There are 3 other instances that required over 600 CPU seconds for the BC algorithm, but were solved to optimality with the BCP algorithm within 141 CPU seconds. Furthermore, the BCP algorithm failed to solve 3 instances of set P, namely, P-n101-k4-C51-V2, P-n76-k4-C38-V2, and P-n76-k5-C38-V3, whereas the BC algorithm solved them within 169 CPU seconds.

The root node lower bound obtained by the BCP algorithm is systematically tighter than those reported for the BC algorithm. For 59% of instances, the lower bound obtained by the BCP algorithm is optimal. Moreover, the average duality gap at the root node of the BCP algorithm is 0.3%, compared to 2.64% for the BC algorithm.

Results for $\theta = 3$

For $\theta = 3$, the BCP algorithm has solved to optimality 3 instances that were previously open in the literature. On the other hand, it failed to solve 3 instances that were solved to optimality by the BC algorithm. A summary of our observations for Sets A, B, and P follows.

- All the instances of set A are solved by the BCP algorithm within 356 CPU seconds, whereas the BC algorithm failed to solve 2 of these instances, namely, A-n63-k9-C21-V3 and A-n80-k10-C27-V4, with 2 CPU hours. Further, the BC algorithm solved 3 other instances in Set A in more than 600 CPU seconds.

- All instances in Set B are equally easy for both algorithms and were all solved to optimally under 472 CPU seconds.

- For Set P, the results are again mixed. The BCP algorithm failed to solve 2 instances, namely, P-n76-k4-C26-V2 and P-n76-k5-C26-V2, within the allocated time limit, whereas the BC algorithm solved them within 122 CPU seconds. The BCP algorithm also failed to solve Instance P-n101-k4-C34-V2 was solved by the BC algorithm within 6582 CPU seconds. On the other hand, the BC algorithm failed to solve Instance P-n60-k15-C20-V5 within 2 CPU hours, whereas the BCP algorithm solved it at the root node within 1.7 CPU seconds.

The BCP algorithm exhibits an average duality gap of 0.26% at the root node, enabling optimal solutions to 64% of the instances at the root node itself. In contrast, the average duality gap at root node of the BC algorithm is 4.47% and only 15% of the instances were solved at the root node itself.

It is apparent from the aforementioned observations that the two methodologies "complement" one another from a computational viewpoint. Many instances that were difficult for one algorithm (i.e., requiring a substantial computational effort) were solved by the other in only a few CPU seconds. In particular, for instances that were formerly solved in a split of one CPU second by the BC algorithm of Bektas et al. (2011), casting the problem as a set partitioning formulation that is solved by a BCP algorithm may not be worthwhile. It has been pointed by Fukasawa et al. (2006) and Baldacci et al. (2008) that the tighter lower bounds obtained by solving set partitioning reformulations may not be worth the inherent computational effort that accompanies column generation techniques. This is especially the case for instances that present a high level of degeneracy, causing a slowdown in the CG procedure. For such instances, the BC algorithm is found to perform better. Conversely, the benefit of employing the BCP and generating tighter root-node dual bounds is realized for the instances that were more challenging to the BC algorithm.

2.4.3 Results for Instances with Random Clustering

Table 2.3 compares the performance of the BCP algorithm against the BC algorithm for instances with random clustering using $\theta = 2$. Table 2.4 presents only the performance of the BCP algorithm for $\theta = 3$ as such results were not reported by Bektas et al. (2011). In these two tables, we report the average performance over Sets A, B, and P. We also report, under Column #Opt, the number of instances that were solved to optimality within 7200 CPU seconds time limit. Since the random instances of Bektas et al. (2011) are not publicly available, we generated, for every instance, three randomly clustered instances and reported average performances. As pointed out by Bektas et al. (2011), it is apparent from results in Table 2.3 that instances with random clustering are significantly more difficult for the BC algorithm than the

Table 2.1: Computational results for $\theta = 2$

	Branch-and-Cut				Branch-and-Cut-and-Price			
Instance	UB	LB-0	BB	CPU	UB	LB-0	BB	CPU
A-n32-k5-C16-V2	519	474.00	1847	113.2	519	516.67	3	38.6
A-n33-k5-C17-V3	451	437.69	38	1.6	451	451.00	1	3.3
A-n33-k6-C17-V3	465	462.12	5	0.7	465	465.00	1	1.4
A-n34-k5-C17-V3	489	486.36	5	0.8	489	489.00	1	3.6
A-n36-k5-C18-V2	505	480.21	612	31.5	505	505.00	1	21.4
A-n37-k5-C19-V3	432	430.40	1	0.8	432	432.00	1	23.5
A-n37-k6-C19-V3	584	559.04	264	28.2	584	584.00	1	6.4
A-n38-k5-C19-V3	476	463.22	27	3.0	476	476.00	1	5.5
A-n39-k5-C20-V3	557	530.58	544	45.6	544	544.00	1	11.5
A-n39-k6-C20-V3	544	525.80	42	4.9	608	608.00	1	6.8
A-n44-k6-C22-V3	608	572.33	210	23.2	608	608.00	1	6.9
A-n45-k6-C23-V4	613	595.67	112	6.8	613	608.40	9	35.4
A-n45-k7-C23-V4	674	630.86	3184	1465.2	674	663.21	53	141.2
A-n46-k7-C23-V4	593	573.95	43	10.2	593	591.58	3	28.9
A-n48-k7-C24-V4	667	630.35	1829	299.8	667	654.12	69	147.1
A-n53-k7-C27-V4	603	589.48	40	15.9	603	603.00	1	69.1
A-n54-k7-C27-V4	690	665.31	372	68.3	690	690.00	1	35.9
A-n55-k9-C28-V5	699	668.04	577	82.6	699	699.00	1	13.2
A-n60-k9-C30-V5	769	750.75	215	75.6	769	769.00	1	48.2
A-n61-k9-C31-V5	638	621.03	243	43.7	638	635.50	5	56.5
A-n62-k8-C31-V4	740	722.34	210	122.7	740	740.00	1	181.2
A-n63-k10-C32-V5	801	759.56	5430	4355.2	801	794.04	29	215.9
A-n63-k9-C32-V5	-	864.77	4749	7200.1	912	907.00	11	274.5
A-n64-k9-C32-V5	763	733.94	1831	1204.3	763	763.00	1	333.2
A-n65-k9-C33-V5	682	665.09	54	29.0	682	681.22	3	82.5
A-n69-k9-C35-V5	680	648.02	2569	817.9	680	672.49	29	363.6
A-n80-k10-C40-V5	998	916.09	4487	7200.0	997	984.22	1	7200.0
B-n31-k5-C16-V3	441	441.00	0	0.1	441	441.00	1	3.0
B-n34-k5-C17-V3	472	472.00	0	0.1	472	472.00	1	15.4
B-n35-k5-C18-V3	626	626.00	0	0.1	626	626.00	1	33.2
B-n38-k6-C19-V3	451	450.82	3	0.7	451	451.00	1	20.1
B-n39-k5-C20-V3	357	356.50	2	0.2	357	357.00	1	72.9
B-n41-k6-C21-V3	481	472.19	79	2.6	481	481.00	1	14.3
B-n43-k6-C22-V3	483	472.11	82	9.2	483	481.86	3	91.3
B-n44-k7-C22-V4	540	537.08	17	3.3	540	540.00	1	26.3
B-n45-k5-C23-V3	497	496.62	5	0.6	497	497.00	1	183.3
B-n45-k6-C23-V4	478	466.72	717	53.7	478	474.50	23	194.7
B-n50-k7-C25-V4	449	446.29	23	0.6	449	449.00	1	109.9
B-n50-k8-C25-V5	916	890.86	7180	3249.2	916	912.14	11	94.4
B-n51-k7-C26-V4	651	650.51	4	0.4	651	651.00	1	141.3
B-n52-k7-C26-V4	450	450.00	0	0.1	450	450.00	1	277.1
B-n56-k7-C28-V4	486	483.44	18	3.0	486	486.00	1	389.5
B-n57-k7-C29-V4	751	748.43	21	1.8	751	751.00	1	370.2
B-n57-k9-C29-V5	942	933.43	115	22.0	942	942.00	1	53.8
B-n63-k10-C32-V5	816	806.70	75	12.2	816	809.00	19	635.4
B-n64-k9-C32-V5	509	507.80	3	0.8	509	509.00	1	837.8
B-n66-k9-C33-V5	808	802.43	34	14.4	808	808.00	1	395.1
B-n67-k10-C34-V5	673	663.37	229	35.8	673	667.11	41	790.7
B-n68-k9-C34-V5	704	700.92	27	9.2	704	704.00	1	1395.9
B-n78-k10-C39-V5	803	791.44	570	248.2	803	803.00	1	726.6
P-n101-k4-C51-V2	455	442.87	647	169.2	470	-	1	7200.0
P-n16-k8-C8-V5	239	239.00	0	0.0	239	239.00	1	0.2
P-n19-k2-C10-V2	147	147.00	0	0.0	147	147.00	1	0.7
P-n20-k2-C10-V2	154	154.00	0	0.0	154	154.00	1	1.2
P-n21-k2-C11-V2	160	160.00	0	0.0	160	160.00	1	3.7
P-n22-k2-C11-V2	162	160.65	3	0.1	162	162.00	1	4.4
P-n22-k8-C11-V5	314	314.00	0	0.0	314	313.00	3	0.0
P-n23-k8-C12-V5	312	303.10	17	0.8	312	312.00	1	0.0
P-n40-k5-C20-V3	294	284.32	29	2.1	294	294.00	1	31.4
P-n45-k5-C23-V3	337	330.84	16	2.2	337	337.00	1	103.9
P-n50-k10-C25-V5	410	377.97	2715	1162.9	410	407.36	9	16.5
P-n50-k7-C25-V4	353	337.11	387	26.7	353	350.49	13	75.2
P-n50-k8-C25-V4	-	342.80	9514	7200.1	392	386.00	17	101.1
P-n51-k10-C26-V6	427	405.14	213	38.8	427	427.00	1	1.6
P-n55-k10-C28-V5	415	387.72	4623	1536.7	415	411.94	13	79.3
P-n55-k15-C28-V8	-	508.65	4537	7200.1	555	555.00	1	1.4
P-n55-k7-C28-V4	361	342.69	967	125.2	361	355.44	21	173.5
P-n55-k8-C28-V4	361	347.79	359	38.9	361	359.22	13	134.6
P-n60-k10-C30-V5	-	406.04	7048	7200.1	445	434.91	139	372.2
P-n60-k15-C30-V8	-	522.22	5122	7200.2	565	564.75	3	4.3
P-n65-k10-C33-V5	487	461.28	2951	1805.5	487	485.22	7	140.3
P-n70-k10-C35-V5	485	468.80	405	175.8	485	485.00	1	71.1
P-n76-k4-C38-V2	383	374.86	108	25.8	411	-	1	7200.0
P-n76-k5-C38-V3	405	396.56	108	16.2	409	-	1	7200.0

Table 2.2: Computational results for $\theta = 3$

	Branch-and-Cut				Branch-and-Cut-and-Price			
Instance	UB	LB-0	BB	CPU	UB	LB-0	BB	CPU
A-n32-k5-C11-V2	386	380.33	5	0.1	386	382.33	3	1.4
A-n33-k5-C11-V2	315	306.80	7	0.5	315	315.00	1	0.7
A-n33-k6-C11-V2	370	355.12	23	1.2	370	370.00	1	0.2
A-n34-k5-C12-V2	419	408.14	26	1.7	419	415.50	5	1.9
A-n36-k5-C12-V2	396	367.30	81	1.3	396	385.63	17	15.5
A-n37-k5-C13-V2	347	344.43	3	0.7	347	347.00	1	16.0
A-n37-k6-C13-V2	431	390.83	309	19.4	431	431.00	1	0.8
A-n38-k5-C13-V2	367	362.94	3	0.7	367	367.00	1	0.5
A-n39-k5-C13-V2	364	331.71	150	4.6	403	403.00	1	3.9
A-n39-k6-C13-V2	403	388.92	5	1.2	403	403.00	1	2.0
A-n44-k6-C15-V2	503	448.92	2019	323.7	503	503.00	1	2.3
A-n45-k6-C15-V3	474	449.68	46	2.9	474	474.00	1	2.6
A-n45-k7-C15-V3	475	451.43	69	7.4	475	475.00	1	3.1
A-n46-k7-C16-V3	462	424.22	349	22.7	462	461.50	5	11.0
A-n48-k7-C16-V3	451	421.72	304	19.0	451	451.00	1	7.0
A-n53-k7-C18-V3	440	417.52	85	5.9	440	440.00	1	46.2
A-n54-k7-C18-V3	482	441.93	430	57.4	482	482.00	1	10.7
A-n55-k9-C19-V3	473	453.69	72	14.1	473	473.00	1	5.5
A-n60-k9-C20-V3	595	543.49	2884	885.2	595	593.50	3	40.5
A-n61-k9-C21-V4	473	445.40	160	14.5	473	473.00	1	8.0
A-n62-k8-C21-V3	596	556.00	2532	859.6	596	594.77	3	57.5
A-n63-k10-C21-V4	593	550.22	1541	279.7	593	592.32	5	13.8
A-n63-k9-C21-V3	-	578.91	8483	7200.1	642	636.33	9	67.4
A-n64-k9-C22-V3	536	516.09	79	22.4	536	536.00	1	146.7
A-n65-k9-C22-V3	500	465.19	174	21.9	500	500.00	1	9.9
A-n69-k9-C23-V3	520	464.76	10201	4752.4	520	520.00	1	20.6
A-n80-k10-C27-V4	-	629.97	4813	7200.1	710	709.27	3	355.3
B-n31-k5-C11-V2	356	355.92	2	0.2	356	354.50	7	1.8
B-n34-k5-C12-V2	369	369.00	0	0.0	369	369.00	1	3.8
B-n35-k5-C12-V2	501	500.74	1	0.2	501	501.00	1	1.6
B-n38-k6-C13-V2	370	362.76	33	1.3	370	370.00	1	1.1
B-n39-k5-C13-V2	280	280.00	0	0.0	280	280.00	1	26.4
B-n41-k6-C14-V2	407	402.72	14	1.0	407	407.00	1	7.9
B-n43-k6-C15-V2	343	343.00	0	0.6	343	343.00	1	7.5
B-n44-k7-C15-V3	395	388.43	41	1.5	395	394.33	5	7.5
B-n45-k5-C15-V2	410	409.25	6	0.9	410	410.00	1	25.5
B-n45-k6-C15-V2	336	332.35	24	4.8	336	336.00	1	5.5
B-n50-k7-C17-V3	393	393.00	0	0.2	393	393.00	1	32.7
B-n50-k8-C17-V3	598	581.34	250	29.4	598	598.00	1	8.0
B-n51-k7-C17-V3	511	510.87	4	0.4	511	511.00	1	10.8
B-n52-k7-C18-V3	359	359.00	0	0.0	359	359.00	1	20.8
B-n56-k7-C19-V3	356	342.98	656	23.5	356	355.50	7	317.4
B-n57-k7-C19-V3	558	558.00	0	0.9	558	558.00	1	42.1
B-n57-k9-C19-V3	681	664.30	2699	471.6	558	558.00	1	42.8
B-n63-k10-C21-V3	599	591.23	65	11.3	599	599.00	1	15.0
B-n64-k9-C22-V4	452	448.37	26	2.4	452	452.00	1	17.7
B-n66-k9-C22-V3	609	585.52	1063	103.5	609	593.00	77	445.4
B-n67-k10-C23-V4	558	551.24	72	7.2	558	551.00	43	96.3
B-n68-k9-C23-V3	523	507.79	1250	110.0	523	523.00	1	98.4
B-n78-k10-C26-V4	606	601.06	11	8.5	606	606.00	1	32.8
P-n101-k4-C34-V2	370	344.87	13879	6581.8	402	-	1	7200.0
P-n16-k8-C6-V4	170	170.00	0	0.0	170	170.00	1	0.2
P-n19-k2-C7-V1	111	111.00	0	0.0	111	111.00	1	0.1
P-n20-k2-C7-V1	117	113.81	7	0.2	117	117.00	1	0.2
P-n21-k2-C7-V1	117	115.69	1	0.2	117	117.00	1	0.2
P-n22-k2-C8-V1	111	111.00	0	0.1	111	111.00	1	0.6
P-n22-k8-C8-V4	249	249.00	0	0.1	249	249.00	1	0.0
P-n23-k8-C8-V3	174	174.00	0	0.1	174	174.00	1	0.0
P-n40-k5-C14-V2	213	208.35	12	1.1	213	212.00	3	10.3
P-n45-k5-C15-V2	238	210.76	230	11.1	238	237.67	3	16.8
P-n50-k10-C17-V4	292	277.41	40	5.0	292	292.00	1	0.8
P-n50-k7-C17-V3	261	246.92	110	6.4	261	259.00	9	57.7
P-n50-k8-C17-V3	262	248.55	53	7.4	262	262.00	1	1.3
P-n51-k10-C17-V4	309	272.36	1483	117.6	309	305.50	11	5.3
P-n55-k10-C19-V4	301	280.48	217	18.1	301	301.00	1	4.1
P-n55-k15-C19-V6	378	350.60	195	36.0	378	378.00	1	0.1
P-n55-k7-C19-V3	271	243.81	819	78.2	271	267.00	15	92.4
P-n55-k8-C19-V3	274	247.11	580	53.6	274	271.29	13	75.3
P-n60-k10-C20-V4	325	298.82	227	282.7	325	320.67	27	24.9
P-n60-k15-C20-V5	-	337.95	11507	7200.0	382	381.55	3	1.7
P-n65-k10-C22-V4	372	338.33	4237	1028.2	372	369.39	7	25.5
P-n70-k10-C24-V4	385	354.46	5541	1468.3	385	382.52	17	87.0
P-n76-k4-C26-V2	309	287.90	840	122.5	309	-	1	7200.0
P-n76-k5-C26-V2	309	287.03	561	90.1	309	-	1	7200.0

Table 2.3: Effect of random clustering on difficulty of instances for $\theta = 2$

	Branch-and-Cut						Branch-and-Cut-and-Price					
	Base			Random			Base			Random		
Type	CPU	BB	#Opt	CPU	BB	#Opt	CPU	BB	#Opt	CPU	BB	#Opt
A	132.7	530.1	13/13	487.9	1504.9	13/13	23.5	5.8	13/13	176.6	18.7	13/13
B	7.1	90.5	10/10	5653.1	10175.4	2/10	23.5	5.8	10/10	1111.3	10.3	8/10
P	0.6	7.2	9/9	1.5	21.4	9/9	22.9	1.2	9/9	62.9	3.0	9/9

Table 2.4: Effect of random clustering on difficulty of instances for $\theta = 3$

	Branch-and-Cut-and-Price					
	Base			Random		
Type	CPU	BB	#Opt	CPU	BB	#Opt
A	3.9	2.7	13/13	73.0	4.5	13/13
B	8.9	2.0	10/10	256.9	2.6	10/10
P	8.9	2.0	9/9	46.3	2.1	9/9

base benchmark instances generated using the clustering technique of Fischetti et al. (1997). It is particularly noteworthy for instances in set B. Whereas these instances were easy for the BC algorithm (see Tables 2.1 and 2.2), the BC algorithm could solve to optimality only 2/10 of the randomly generated ones. Instances with random clustering are also more challenging for the BCP algorithm, though one could argue that there is a less marked difficulty for the latter. In fact, the BCP had greater success with instances in Set B and solved 8/10 instances to optimality. For instances constructed with $\theta = 3$, the results in Table 2.4 highlight the difficulty caused by randomly clustering customers. However, all such instances were solved to optimality using the BCP algorithm, despite the relative CPU time increase compared to their base counterparts.

2.5. Conclusion

This chapter develops an exact solution approach to the GVRP, a variant of the vehicle routing problem where customers are partitioned into clusters and exactly one customer in each cluster must be visited by a single route, subject to route capacity constraints. The proposed branch-and-cut-and-price algorithm yields promising computational results for benchmark instances. In particular, it has solved to optimality eight instances from the literature that were otherwise open to date. The algorithm employs, at its heart, a dynamic programming-based approach to solve the column generation pricing subproblem.

The computational performance of the branch-and-cut-and-price algorithm has been compared against that of a state-of-the-art branch-and-cut algorithm (Bektas et al. 2011). In our experience, no algorithm consistently outperformed the other over the considered testbed of benchmark instances, which comprised instances where customers are clustered either based on proximity considerations or using a random clustering scheme. What is perhaps more remarkable is that the two algorithms tend to "complement one another" computationally in the sense that many instances that were harder to solve using the branch-and-cut algorithm were efficiently solved using the proposed branch-and-cut-and-price algorithm, and vice versa. Further, for instances with random customer clustering, the proposed branch-and-cut-and-price algorithm was found to perform overall better than the branch-and-cut algorithm. Whereas the incorporation of rounded capacity inequalities was found to be most useful for the branch-and-cut-and-price algorithm, it would be worthwhile to further identify other computationally beneficial types of cuts that can enhance both branch-and-cut and branch-and-cut-and-price algorithms. The single instance that remains unsolved by these two methodologies to date, namely Instance A-n80-k10-C40-V5, can serve as a motivation for such future developments.

CHAPTER 3

A MULTI-START OPTIMIZATION-BASED HEURISTIC FOR A FOOD BANK DISTRIBUTION PROBLEM

In this chapter, we investigate a variant of the Vehicle Routing-Allocation Problem that arises in the distribution of pallets of goods by a food bank to a network of relatively distant nonprofit organizations. Vehicles are routed to selected intermediate delivery sites to which the nonprofit organizations travel to collect their demand. The logistical cost is shared and the objective is to minimize a weighted average of the food bank vehicle routing cost and the travel cost of the nonprofit organizations. We develop an efficient multi-start heuristic that iteratively constructs initial solutions to this problem and subsequently explores their neighborhoods via local improvement and perturbation schemes. In our experience, the proposed heuristic substantially outperforms alternative optimization-based heuristics in the literature in terms of the solution quality and computational efficiency and consistently yields solutions with an optimality gap of 0.5% on average.

3.1. Introduction and Motivation

"We have the means; We have the capacity to eliminate hunger from the face of the earth in our lifetime. We need only the will." (John F. Kennedy, 1963)

Poverty, hunger, malnutrition, and disease are linked in a vicious cycle that continues to affect millions of lives worldwide. While there is enough food to entirely sustain the world population, over 800 million people have inadequate access to food and are undernourished (Uvin 1994; Soubbotina 2004). Although more severe in developing nations, the devastating effect of hunger causes great societal concern in high-income nations as well (Feeding America 2015). The British Medical Journal indicated in 2013 that hunger in the United Kingdom had reached a state of "public health emergency." Likewise, recent statistics suggest that 17.6 million households – nearly 1 in 7 people – in the USA are food insecure (World Hunger 2015). Food banks and other nonprofit organizations play an important role in collecting and distributing goods to needy individuals directly or via local agencies that serve local communities. This can include a large network of shelters, food pantries, and soup kitchens that are supported by food banks. Due to the recurrent demand by these local agencies and their geographical spread, food bank delivery operations need efficient planning in order to reduce the associated logistical cost. We consider in particular the Vehicle Routing with Demand Allocation Problem (VRDAP) introduced in Ghoniem et al. (2013) in the context of food bank distribution operations.

Ghoniem et al. (2013) compared two optimization-based techniques for the VRDAP: (i) A relax-and-fix heuristic with symmetry-defeating constraints and (ii) a column generation (CG) heuristic with a compalmentary column generation feature (Ghoniem and Sherali, 2009). However, the CG approach exhibited a well-known tailing-off effect and both methodologies became computationally onerous for instances involving up to 10 delivery sites and 50 customers. Further, Solak et al. (2014) tackled instances of the VDRAP with up to 25 delivery sites and 50 customers using a classical Benders decomposition heuristic and a logic-based Benders decomposition. Similarly, although the reported results compared favorably against CPLEX within a time limit of one CPU hour, both Benders decomposition approaches exhibited a slow convergence.

The objective of this chapter is to develop an optimization-based heuristic that overcomes the limitations of CG and Benders decomposition approaches by improving upon the solution quality and significantly reducing the computational effort. Solving the VRDAP involves the optimization of the following intertwined decisions: (i) Selecting a subset of delivery sites from a set of candidate locations; (ii) assigning every customer to a selected delivery site; and (iii) routing vehicles to supply customers at their designated delivery sites. The integrated nature of the problem makes two-step heuristics (of the type assign customers first, route second) particularly ineffective. In fact, an enhancement in the customer assignment cost is often accompanied by an increase in the routing cost, and vice versa. Consequently, this also limits the applicability of classical VRP heuristics to the VRDAP. Although optimization decomposition techniques based on CG or Benders decomposition capture the integrated nature of the problem, their slow convergence constitutes a disadvantage.

This chapter proposes a multi-start heuristic for VRDAP that yields near-optimal solutions (with a 0.5% optimality gap on average) within a few CPU seconds, as opposed to one CPU hour for decomposition-based heuristics in the literature. The heuristic overcomes the challenge posed by this integrated problem by coordinating two complementary subproblems and iteratively exploring better routing and customer assignment decisions. At each start of the heuristic, a new allocation of customers to delivery sites is enforced, thereby reducing the problem to a capacitated VRP (CVRP) which is subsequently solved using a CVRP heuristic.

The remainder of this chapter is organized as follows. Section 3.2 presents the overall heuristic and the initialization procedure it employs. Local search and perturbation procedures that enable the exploration of solution neighborhoods are described in Section 3.3. Computational results are discussed in Section 3.4 and Section 3.5 concludes this chapter with a summary of our findings.

3.2. Multi-Start Heuristic

In this section, we introduce our notation along with a formal problem statement. This is followed by an overall description of the proposed heuristic and a detailed description of the initialization procedure.

3.2.1 Notation and Problem Statement

We consider a set V of identical vehicles having a capacity of Q (pallets). All the vehicles are initially located at a central depot, denoted by node 0. Any vehicle tour starts at the depot, sequentially visits a subset of delivery sites selected from a set of candidate locations, S, where it delivers a load of food pallets, and returns to the depot. Following Ghoniem et al. (2013) and Solak et al. (2014), it is assumed that during any planning instance any delivery site can be used at most once. That is, it can be included in no more than one vehicle tour. Because delivery sites are not owned by the food bank, but rather by retailers or religious organizations that give access to their parking lots for charitable purposes, this assumption limits and balances the use of these facilities. Each customer $k \in K$ expresses a demand of d_k pallets which must be picked up from a designated delivery site. The VRDAP aims at constructing a maximum of $|V|$ tours that minimize the weighted average of the vehicle routing and customer assignment/travel costs. Our notation is summarized as follows:

- S: Set of candidate delivery sites.
- $N \equiv S \cup \{0\}$: Set of delivery sites and central depot.
- K: Set of customers.
- V: Set of vehicle tours.
- A: Set of all arcs $(i, j), i, j \in N, i \neq j$, that can be included in a vehicle tour.

- c_{ij}: Cost of arc $(i, j) \in A$.
- f_{kj}: Cost for customer k to travel to delivery site j, $\forall k \in K, j \in N$.
- d_k: Demand of customer $\forall k \in K$.
- Q: Vehicle capacity.

We also assign a weight of λ and (1 - λ) for the vehicle routing and customer assignment costs, respectively, where $\lambda \in [0, 1]$. To place equal or greater emphasis on either cost, every instance in our computational study is run using λ = 0.25, 0.5, or 0.75.

3.2.2 Overall Algorithm

To overcome the challenges posed by the interdependence of vehicle routing and customer assignment decisions in the VRDAP a multi-start scheme is employed along with local search procedures and perturbation mechanisms which enable a good exploration of the feasible space and solution neighborhoods. The proposed heuristic rests on the following two observations: (i) Under fixed routing decisions, it is possible to obtain a solution completion by optimally assigning customers to delivery sites along the pre-specified routes using a generalized assignment problem and (ii) conversely, under a given assignment of customers to chosen delivery sites, the problem reduces to a CVRP which can be solved to optimality or heuristically.

The overall scheme of the heuristic is provided in Algorithm 2. First, an initial solution, denoted by $\bar{\pi}$, is constructed. Thereafter, in each iteration in the main loop of Algorithm 2 (lines 5-28), a new starting solution is initiated using Procedure START($\bar{\pi}$) by making random alterations to the initial solution $\bar{\pi}$. The resulting solution π^o is subsequently refined in the inner loop of Algorithm 2 (lines 9-22) using a series of local improvement and perturbation procedures, denoted by CVRP(π), GAP(π) and SITEREMOVAL(π). Procedure CVRP(π) targets routing decision improvements,

Algorithm 2 MULTI-START HEURISTIC FOR VRDAP

```
1: Comment: $\omega(\pi)$ is the cost of solution $\pi$
2: $idle^o \leftarrow 0$; $idle^* \leftarrow 0$
3: Construct an initial solution $\bar{\pi}$ using Procedure INITIALIZE()
4: $\pi^* \leftarrow \bar{\pi}$
5: while $idle^* < \Delta^*$ do
6:     Create an altered starting solution $\pi^o$ by invoking Procedure START($\bar{\pi}$)
7:     $\pi \leftarrow \pi^o$
8:     $idle^o \leftarrow 0$
9:     while $idle^o < \Delta^o$ do
10:        Refine routing decisions using the probabilistic Procedure CVRP($\pi$)
11:        Refine customer assignment decisions using Procedure GAP($\pi$)
12:        Delete subset of delivery sites and re-allocate customers using Procedure SITEREMOVAL($\pi$)
13:        $idle^o \leftarrow idle^o + 1$
14:        if $\omega(\pi) < \omega(\pi^o)$ then
15:            $idle^o \leftarrow 0$
16:            $\pi^o \leftarrow \pi$
17:        end if
18:        if $idle^o \neq 0$ and $idle^o \ mod \ \xi = 0$ then
19:            $\pi \leftarrow \pi^o$
20:            Perturb solution randomly using Procedure PERTURB($\pi$) to escape local optima
21:        end if
22:    end while
23:    $idle^* \leftarrow idle^* + 1$
24:    if $\omega(\pi^o) < \omega(\pi^*)$ then
25:        $idle^* \leftarrow 0$
26:        $\pi^* \leftarrow \pi^o$
27:    end if
28: end while
```

under fixed customer assignments to delivery sites. Conversely, Procedure GAP(π) seeks to refine customer-delivery site assignments by solving a generalized assignment problem, under fixed routing decisions. At last, Procedure SITEREMOVAL(π) considers the removal of a subset of delivery sites and the re-allocation of their associated customers. Moreover, a perturbation scheme is triggered at the end of the inner loop in order to escape from local optima if the solution at hand did not improve over multiple consecutive iterations. Further, if the algorithm stops if it fails to identify an improved solution over Δ^* consecutive iterations/starts. In what follows, the initialization procedure is discussed in Section 3.2.3 and the local improvement and perturbation schemes are presented in Section 3.3.

3.2.3 Initialization Procedure

The first step in Algorithm 1 is to construct an initial solution. This is achieved using Procedure INITIALIZE() which operates in three steps. First, customers are assigned to delivery sites using a generalized assignment problem (GAP) in a manner that minimizes the overall customer travel cost, subject to a capacity constraint for any delivery site. The latter capacity constraint is implied by the vehicle capacity Q; if a delivery site were assigned a total demand that exceeds Q, no vehicle would be able to serve it, resulting in infeasibility. Second, under fixed customer assignment decisions, a solution completion in the routing decisions can be obtained by solving the corresponding CVRP heuristically. If the second step failed, a recovery step is triggered whereby routes from the second step are fixed and customer assignments are optimized accordingly. The three steps are detailed next.

3.2.3.1 Customer Assignment

Formally, let t_{ks} be a binary variable such that $t_{ks} = 1$ if and only if customer k is assigned to delivery site s with an associated travel cost f_{ks}. An optimized assignment of customers to delivery sites is obtained by solving the following generalized assignment problem, denoted by **Model GAP1**:

$$\textbf{GAP1:}\ \text{Minimize} \sum_{k\in K}\sum_{s\in S} f_{ks}t_{ks} \tag{3.1a}$$

$$\text{subject to} \sum_{s\in S} t_{ks} = 1 \qquad \forall k \in K \tag{3.1b}$$

$$\sum_{k\in K} t_{ks}\, d_k \leq Q \qquad \forall s \in S \tag{3.1c}$$

$$t \text{ binary.} \tag{3.1d}$$

The objective function (A.1) minimizes the total customer travel cost. Constraint (A.2) ensures that every customer is assigned to exactly one delivery site. Constraint (A.3) guarantees that the total customer demand allocated to any delivery site does not exceed the vehicle capacity Q. Constraint (A.4) specifies that the t-variables are binary. If GAP is infeasible for a given instance, it is concluded that this instance itself is infeasible and the algorithm terminates. Furthermore, despite the

NP-hardness of GAP in general, Model GAP1 solves in a split of a second for all instances in our testbed using CPLEX 12.6. In the event some problem instances involved computationally challenging GAPs, one could consider solving them using the branch-and-price algorithm of Savelsbergh (1997) or a competitive heuristic.

3.2.3.2 Vehicle Routing

Upon fixing customer assignments to delivery sites in the previous step, the problem reduces to a CVRP problem in which the demand of each delivery site is the total demand of customers assigned to it. Specifically, we solve a CVRP over the set of all delivery sites that have at least one assigned customer, denoted by S^+. The purpose of this step is to partition the sites in S^+ into no more than $|V|$ vehicle routes. Any route starts at the depot and sites from S^+ are iteratively added to it. To illustrate, let $r = (0, ..., v)$ be a partial route at hand and $\tilde{S}^+$ be the set of all sites in S^+ that can be individually added to r without violating the vehicle capacity. If $\tilde{S}^+$ is empty, we follow the above procedure with a new route. Otherwise, we select from $\tilde{S}^+$ a site that is closest to the last site in the partial route (node v). Such a node is removed from S^+ and added to r. This algorithm terminates successfully when S^+ is empty (and no more than $|V|$ routes have been formed). Otherwise, if certain delivery sites could not be successfully assigned to any of the $|V|$ routes, the next step is invoked.

3.2.3.3 Solution Recovery

If the previous step failed, a solution recovery strategy is triggered to ensure that all delivery sites in S^+ are used and all the customers are assigned without violating the capacity of vehicles. To this end, we first consider a demand of 0 for the delivery sites of S^+ and insert them in their minimum cost insertion point in the $|V|$ tours at hand. This provides a partitioning of all the sites in S^+ into $|V|$ routes. The routes are fixed and all customers are assigned anew to the existing tours in a manner that minimizes their total travel cost, subject to route capacity constraints, using a generalized assignment problem (similar to the one presented earlier in this section). Note that if the problem instance at hand is feasible, then solving this generalized assignment problem will necessarily yield an initial feasible solution. Otherwise, if the solution recovery step fails, the problem instance itself must be infeasible and the overall algorithm breaks.

3.3. Local Improvement & Perturbation Schemes

In this section, we describe the local improvement and perturbation schemes that contribute to intensifying and diversifying the search in the proposed multi-start heuristic.

3.3.1 Procedure Start

Because the initial customer assignment decisions may not guarantee global optimality, the proposed algorithm iteratively alters the initial solution. Specifically, the customer assignment decisions are altered and a new routing completion is sought. At each start of Algorithm 1 (line 6), an altered customer assignment is considered and is further refined in the inner loop of Algorithm 1 (lines 9-22), thereby ensuring a diverse exploration of the feasible space.

Procedure Start($\bar{\pi}$) is grounded in the initial solution $\bar{\pi}$ obtained from Procedure Initialize(). Note that because the initial solution is based on solving a generalized assignment problem (Model GAP1) that minimizes the total travel cost of customers to delivery sites, it tends to include numerous delivery sites in $\bar{\pi}$. Therefore, in each start, procedure Start($\bar{\pi}$) randomly removes ψ delivery sites from $\bar{\pi}$. The parameter ψ itself is randomly selected with equal probability from the set $Z = \{0, 1, \ldots \kappa * |S^+|\}$, where S^+ is the set of all delivery sites visited by the solution $\bar{\pi}$ and $\kappa \in (0, 1]$ is an algorithm parameter (that we set to 0.25 in our computational study). In each of the ψ iterations, one delivery site s^- is randomly removed from S^+ using the following scheme:

- with probability of 0.25, s^- is the delivery site in S^+ with lowest aggregate demand (ties are broken arbitrarily).
- with probability of 0.25, s^- is the delivery site in S^+ with second lowest aggregate demand.
- with probability of 0.5, s^- is the delivery site with i^{th} lowest aggregate demand where i is randomly selected from $\{3, 4, \ldots, \psi + 2\}$.

After ψ iterations, the remaining delivery sites of S^+ are used to form an initial solution which will be further improved in the inner loop of algorithm. Using the revised/reduced set S^+, a solution is constructed following the scheme of Procedure Initialize(). Specifically, a generalized assignment problem (Model GAP1) is solved whereby the assignment of customers to delivery sites out of the set S^+ is discouraged using a large penalty.

3.3.2 CVRP Heuristic

Procedure Cvrp(π) improves the routing decision of the VRDAP solution at hand, π, by fixing the customer assignments and solving the resulting CVRP problem. Each delivery site $s \in S$ can be viewed as a "CVRP customer" having the aggregate demand of its assigned customers. When distances satisfy the triangle inequality, as in our instances, delivery sites with zero demand can be removed from the resulting CVRP problem. This CVRP problem is solved following the heuristic of De Franceschi et al. (2006). An integer programming-based (IP) refinement procedure lies at the heart of this heuristic and involves the following three main steps:

- **Solution destruction.** Let $R = (0, s_1, \ldots, s_{|R|}, 0)$ be a route in π that starts and ends at depot (node 0). With probability of 0.5, we extract all the delivery sites s_i for which i is not a divisor of 3 (i.e. $(s_1, s_2, s_4, s_5, s_7 \ldots)$. Otherwise, with probability of 0.5, all delivery sites s_i for which $i \, mod \, 3 \neq 1$ are extracted from R (i.e. $s_2, s_3, s_5, s_6, s_8, \dots$). This procedure is applied to all the routes of π and any two consecutive remaining delivery sites are connected to each other resulting in a destructed solution.

- **Column generation.** The purpose of this step is to construct sequences of delivery sites that can be inserted in the destructed solution. For this purpose, we define an insertion point p as an edge newly formed in π due to extraction of delivery sites. The set of extracted delivery sites associated with insertion point p is called o_p. For each insertion point p, we define set σ_p which contains members of o_p as well as the μ nearest delivery sites to members of o_p that are also extracted from π. Then, we define H_p as the set of all subsets of σ_p that have cardinality of one, two, or three. For each member of H_p, we construct a sequence of the associated delivery sites that would result in a minimum insertion cost if inserted in p.

- **Reallocation IP model.** The purpose of this step is to repair the destructed solution by inserting the selected sequences generated in previous step into insertion points. To this end, an IP model is solved with the objective of minimizing the insertion cost of inserted sequences while assuring that the resulting solution is a feasible CVRP solution. Note that the resulting IP model is always feasible and its optimal solution is at least as good as π because one of the generated sequences of each insertion point p is the sequence of delivery sites that was originally extracted from it.

3.3.3 Procedure GAP: Optimal Assignment of Customers to Routes

Procedure GAP(π) considers an input solution π comprising $R = \{r_1, ..., r_{|R|}\}$ fixed routes and seeks an optimized assignment of customers to these routes. To this end, a generalized assignment problem similar to Model GAP1 is solved whereby each customer must be assigned to one of the existing $|R|$ routes, subject to vehicle capacity constraints. Here, the cost of assigning a customer k to a route r equals $\min\{f_{ks}|s \in r\}$. This generalized assignment problem was solved to optimality for instances in our testbed in a fraction of one CPU second, often at the root-node of the B&B/C algorithm of CPLEX 12.6 and, hence, causes no computational burden in our experience.

3.3.4 Delivery Site Removal

Procedure SiteRemoval(π), delineated in Algorithm 3, seeks to improve an input solution π by deleting certain delivery sites and reallocating their customers. For each site s visited by π, we define rs_s as the estimated savings resulting from removing s from π and connecting its predecessor and successor. The estimated increase in assignment cost due to the reallocation of customers of s to other sites is denoted by ac_s. Denoting by K_s the customers assigned to s, we let $ac_s = \sum\limits_{k \in K_s} (f_{k,s'_k} - f_{k,s})$, where s'_k is the delivery site in π which is nearest to customer k. The total estimated savings for the removal of s is defined as $ts_s = (\lambda)rs_s - (1-\lambda)ac_s$. Procedure SiteRemoval($\pi$) uses this metric to assess the potential savings due to site removals. It first identifies a site s^* that has the largest ts value and has not yet been considered for removal ($\eta_{s^*} = False$). If s^* has a promising estimated savings, we delete it from the solution and resolve a generalized assignment problem in order to re-assign the customers in K_{s^*} in the updated solution. If this results in a better solution, it is saved and the algorithm proceeds to the next iteration. Otherwise, the previous solution is restored and s^* will not be considered for removal in the next iterations (by setting $\eta_{s^*} \leftarrow True$).

Because ts_s is only an estimated savings, the removal of s is deemed promising if $ts_s \geq 0$ or when it is negative, but its absolute value is small enough compared to the cost of solution π, denoted by $\omega(\pi)$ (i.e., $\frac{|(\lambda)rs_s - (1-\lambda)ac_s|}{\omega(\pi)} < \epsilon$ where ϵ is a small scalar that is set to 0.02 in our computational study). If s is the only site in a route and its removal makes the solution infeasible, it is not considered for removal. The

Algorithm 3 Delivery site removal procedure

```
1: Input: Solution $\pi$
2: Comment: $S^+_\pi$: Set of sites visited by $\pi$.
3: $\eta_s \leftarrow False \quad \forall s \in S^+$
4: $\bar{\pi} \leftarrow \pi$
5: for all $s \in S^+$ do
6:     $rs_s \leftarrow$ Estimated saving in routing cost due to the deletion of $s$
7:     $ac_s \leftarrow$ Estimated increase in assignment cost due to the deletion of $s$
8: end for
9: $s^* \leftarrow \arg\max\{(\lambda)\, rs_s - (1-\lambda)\, ac_s |\ s \in S^+_\pi \textbf{ AND } \eta_s = \text{ False}\}$
10: while $((\lambda) rs_{s^*} - (1-\lambda) ac_{s^*} \geq 0)$ OR $(\frac{|(\lambda) rs_{s^*} - (1-\lambda) ac_{s^*}|}{\omega(\pi)} < \epsilon)$ do
11:     Extract $s^*$ from $\bar{\pi}$
12:     Solve GAP($\bar{\pi}$) to assign customers to the new solution
13:     if $\omega(\bar{\pi}) < \omega(\pi)$ then
14:         Remove site $s^*$: $S^+_\pi \leftarrow S^+_\pi \setminus \{s^*\}$
15:         Update the solution: $\pi \leftarrow \bar{\pi}$
16:     else
17:         Restore previous solution: $\bar{\pi} \leftarrow \pi$
18:         Discard $s^*$ in future iterations: $\eta_{s^*} \leftarrow$ True
19:     end if
20:     $s^* \leftarrow \arg\max\{(\lambda) rs_s - (1-\lambda) ac_s |\ s \in S^+_\pi \textbf{ AND } \eta_s = \text{ False}\}$
21: end while
```

algorithm terminates when there is no site that has a promising t_s value and has not been checked for removal yet.

3.3.5 Solution Perturbation

After each ξ iterations of the inner loop, Procedure Perturb(π) is invoked with the objective of altering the solution at hand, thereby escaping local optima and ensuring a more diversified exploration of the feasible space. Procedure Perturb(π) consists of two steps. In the first step, ρ sites are removed from the current solution π. This task is similar to Procedure SiteRemoval(π) except that selected sites are removed even if their removal negatively affected the quality of the solution. In the second step, several delivery sites are inserted in the solution. Specially, let S^o be the set of all delivery sites not visited by routes in the current solution π (including sites removed in the first step). In the second step, $\rho + \gamma$ sites in S_o are inserted in the solution π based on their estimated cost savings.

For each site $s \in S^o$, let β_s be the routing cost resulting from inserting s in π in a way that incurs a minimum routing cost. Also, let α_s be the estimated savings in assignment cost due to the insertion of s. For each $k \in K$, let s_k be the delivery site in π to which k is assigned and $\bar{K}_s$ be the set of customers that are closer to s than its currently assigned site s_k. Accordingly, α_s is estimated as $\sum_{k \in \bar{K}_s} (f_{k,s_k} - f_{k,s})$ and the net estimated cost savings of inserting $s \in S^o$ is $\tau_s = \alpha_s - \beta_s$. In each of the ρ iterations, a delivery site $s^* \in S^o$ having the largest τ_{s^*} value is inserted in the current solution at its best insertion point. As a consequence, s^* is removed from S^o and a generalized assignment problem is solved in order to assign customers to the new solution before proceeding to the next iteration. The remaining γ delivery sites are added as follows. In each iteration if $\tau_{s^*} \geq 0$, the delivery site s^* is inserted as described above. Otherwise, the delivery site selection is performed probabilistically where the probability of selecting $s \in S^o$ equals $\frac{-1/\tau_s}{\sum\limits_{v \in S^o} -1/\tau_v}$.

3.4. Computational Study

In this section, we assess the computational efficacy of the proposed heuristic and the quality of the solutions it generates. Our heuristic solutions are compared against the solutions obtained by solving the MIP model in the Ghoniem et al. (2013) with a time limit of 3600 CPU seconds. Due to the probabilistic nature of the heuristic, we report its average performance over five independent runs. The heuristic was implemented in C# under Visual Studio and all runs were performed on a Windows 7 professional 64-bit operating system with an Intel Core i7-2600 CPU with 3.40 GHz and 12 GB RAM desktop.

3.4.1 Description of Testbed & Algorithmic Settings

We consider two sets of instances, namely Set A and Set B, comprising 100 randomly generated instances. For each instance, three runs are performed with $\lambda =$ 0.25, 0.5, or 0.75, respectively, where λ and (1-λ) designate the relative weight associated with the routing cost and the customer assignment cost. The size of an instance is characterized by two of its primary parameters: The number of delivery sites, $|S|$, and the number of customers $|K|$. Set A comprises 40 instances with the following instance sizes: $|S| = 10$ or 25 and $|K| = 20$, 30, 40, or 50. Set B comprises 60 instances with $|S| = 35, 40$ or 50 and $|K|$ ranging from 40 to 90. For each of the resulting 20 $(|S|, |K|)$ combinations, five distinct instances were constructed using the data generation scheme in Ghoniem et al. (2013) and Solak et al. (2014) as follows. The coordinates of customers and delivery sites were randomly generated using a uniform distribution in a two-dimensional Euclidean space where customers and delivery sites are 25 to 75 miles away from the food bank depot located at the origin. The demand of each customer is randomly generated between 1 and 5 pallets with the following probabilities: $P(\text{“}d_k = 1\text{”}) = 0.5$, $P(\text{“}d_k = 2\text{”}) = 0.2$, $P(\text{“}d_k = 3\text{”}) = 0.1$, $P(\text{“}d_k = 4\text{”}) = 0.1$, and $P(\text{“}d_k = 5\text{”}) = 0.1$. Each vehicle has a capacity $Q = 25$ (pallets) and the total number of vehicles considered is $\left\lceil (\sum_{k=1}^{|K|} d_k)/Q \right\rceil + 2$.

Table 3.1 summarizes key parameter values in our heuristic that are obtained after extensive computational experiments:

- Δ^* is the maximum number of idle iterations in the main loop of Algorithm 1 (Section 3.2.2).
- Δ^o is the maximum number of idle iterations in the inner loop of Algorithm 1.
- μ is the number of nearest delivery sites considered in the variable generation scheme of Procedure Cvrp(π).
- κ is used to specify the number of extracted delivery sites in Procedure Start (Section 3.3.1).
- ϵ is a tolerance value that determines whether a site can be removed in Procedure SiteRemoval (Section 3.3.4).

- ξ is the frequency at which the solution is perturbed in Algorithm 1.
- ρ and γ specify the number of delivery sites that are removed or added in Procedure Perturb (Section 3.3.5).

Table 3.1: Parameter values in the proposed heuristic

Δ^*	Δ^o	μ	κ	ϵ	ξ	ρ	γ
3	7	$\lceil \lvert S\rvert * 0.4\rceil$	0.25	0.02	2	$\lceil \lvert S\rvert * 0.1\rceil + 2$	$\lceil \lvert S\rvert * 0.1\rceil + 2$

3.4.2 Discussion of Results

The baseline performance of the heuristic is first discussed under equal objective weights $\lambda = 0.5$ for the vehicle routing cost and the customer travel cost. Thereafter, greater emphasis is placed on either of the two costs with $\lambda = 0.25$ or 0.75.

3.4.2.1 Baseline Heuristic Performance

Tables 3.2 and 3.3 report the results obtained for the proposed heuristic with $\lambda = 0.5$ for instances in Set A and Set B, respectively. In Table 3.2, the first two columns specify the instance size and number, respectively. Columns 3-5 report the best feasible solution found by CPLEX within a time limit of one CPU hour, the solver optimality gap at termination, and the CPU time (in seconds). Columns 6-9 report the following for the proposed heuristic: (i) The best solution found over 5 runs; (ii) Gap-UB, the gap between the average heuristic solution over five runs and the best incumbent solution found by CPLEX whereby a negative value indicates a savings achieved by our heuristic over CPLEX; (iii) Gap-LB, the gap between the average heuristic solution and the lower bound obtained using the column generation scheme in Ghoniem et al. (2013); and (iv) the CPU time (in seconds). For instances of Set B, Gap-LB is not reported since computing the aforementioned column generation-based lower bound was computationally impractical. For such instances, we compare the average heuristic solution over five runs to (i) the CPLEX incumbent (Gap-UB) and (ii) the best heuristic solution obtained (Dev.%) to demonstrate the computational consistency of the heuristic.

Table 3.2: Heuristic performance for Set A, $\lambda = 0.5$

		CPLEX			Heuristic			
$(\lvert S\rvert, \lvert K\rvert)$	Inst.	UB	relGap%	CPU	Best	Gap-UB% †	Gap-LB% ‡	CPU
(10, 20)	1	422	0.0	81.4	422	0.00	0.00	4.4
	2	409	0.0	2.5	409	0.00	0.00	1.8
	3	380	0.0	415.1	380	0.05	0.98	10.7
	4	459.5	0.0	277.1	459.5	0.00	0.00	1.7
	5	389.5	0.0	99.2	389.5	0.00	0.00	2.0
(10, 30)	1	534	0.0	355.1	534	0.00	0.09	3.5
	2	553	0.0	17.3	553	0.00	0.00	1.8
	3	490.5	0.0	1164.9	490.5	0.57	0.62	6.7
	4	642	0.0	3121.3	642	0.00	0.00	9.1
	5	567.5	5.7	3600.0	567.5	0.00	1.98	5.4
(10, 40)	1	731	4.4	3600.0	731	0.00	0.00	7.5
	2	732.5	0.0	2084.0	732.5	0.00	0.32	3.6
	3	793.5	5.9	3600.0	793.5	0.00	0.12	4.6
	4	690.5	6.4	3600.0	690.5	0.93	1.72	12.6
	5	737.5	7.1	3600.0	737.5	0.00	0.00	3.1
(10, 50)	1	1014.5	14.6	3600.0	996.5	-1.77	0.43	7.1
	2	914	0.0	2063.6	914	0.00	0.29	7.1
	3	790.5	15.7	3600.0	769.5	-2.66	0.00	4.3
	4	895.5	11.6	3600.0	894.5	-0.11	0.03	10.4
	5	863	8.8	3600.0	863	0.00	0.00	8.3
(25, 20)	1	340	20.7	3600.0	340	0.00	0.00	6.4
	2	336	22.7	3600.0	334	-0.60	0.00	11.1
	3	383	23.0	3600.0	383.5	0.68	0.99	15.5
	4	342.5	21.4	3600.0	341.5	-0.29	0.74	9.7
	5	337.5	20.5	3600.0	337.5	0.56	0.56	7.5
(25, 30)	1	447	16.7	3600.0	447	0.83	0.88	29.1
	2	455	27.8	3600.0	448.5	-1.43	0.08	21.2
	3	485.5	20.2	3600.0	482	-0.33	0.39	21.5
	4	527	25.7	3600.0	527	0.00	0.00	15.1
	5	574.5	43.8	3600.0	491	-14.26	0.70	30.2
(25, 40)	1	623.5	30.9	3600.0	537	-13.87	0.00	15.0
	2	815.5	48.5	3600.0	597	-26.52	0.49	39.9
	3	738.5	46.1	3600.0	580.5	-20.14	1.74	40.6
	4	730.5	32.2	3600.0	626	-14.22	0.50	46.7
	5	651.5	30.5	3600.0	593	-8.95	0.48	41.3
(25, 50)	1	996.5	49.0	3600.0	736	-26.13	0.20	46.9
	2	1053	50.2	3600.0	733	-30.37	0.03	75.6
	3	805.5	42.2	3600.0	661.5	-17.88	0.74	29.2
	4	842	44.0	3600.0	701	-16.75	0.72	42.7
	5	976	52.9	3600.0	721	-25.72	4.04	53.1

†Gap-UB: Gap between average heuristic solution and cplex incumbent after 1 CPU hour.
‡Gap-LB: Gap between average heuristic solution and CG-based LB.

For instances in Set A, CPLEX solved only 11/40 instances to optimality within one CPU hour. For $|S| = 10$, the heuristic solution was obtained in 5.8 CPU seconds with an optimality gap of 0.33% on average. The average heuristic solution was at least as good as the incumbent of CPLEX in 17/20 instances and outperformed CPLEX for 3/20 instances. For $|S| = 25$, the heuristic required about 30 CPU seconds on average and outperformed CPLEX in 15/20 instances with an accompanying

Table 3.3: Heuristic performance for Set B, $\lambda = 0.5$

		CPLEX			Heuristic Performance				
$(\|S\|, \|K\|)$	Inst.	UB	relGap%	CPU	Best	Average	Gap-UB% †	Dev.% ‡	CPU
(35, 40)	1	589.5	31.05	3600.0	552.00	552.00	-6.36	0.00	44.4
	2	662	39.86	3600.0	541.50	546.67	-17.42	0.95	43.4
	3	687.5	47.3	3600.0	543.00	544.67	-20.78	0.31	43.1
	4	756	47.87	3600.0	568.00	572.00	-24.34	0.70	50.4
	5	655.5	39.56	3600.0	566.00	566.00	-13.65	0.00	46.6
(35, 50)	1	868	47.78	3600.0	636.00	636.00	-26.73	0.00	68.6
	2	737	37.7	3600.0	629.50	629.50	-14.59	0.00	48.2
	3	889	47.4	3600.0	674.50	676.33	-23.92	0.27	57.9
	4	764.5	48.88	3600.0	610.00	613.17	-19.80	0.52	55.1
	5	790	51.74	3600.0	634.50	636.67	-19.41	0.34	46.7
(35, 60)	1	806.5	43.91	3600.0	687.00	688.33	-14.65	0.19	75.8
	2	1140.5	55.76	3600.0	741.50	744.50	-34.72	0.40	60.4
	3	952	47.84	3600.0	696.00	698.67	-26.61	0.38	58.6
	4	1227.5	60.38	3600.0	737.50	740.17	-39.70	0.36	60.0
	5	1166.5	54.84	3600.0	795.50	798.50	-31.55	0.38	85.7
(35, 70)	1	1443.5	58.18	3600.0	859.00	864.17	-40.13	0.60	86.4
	2	1238.5	54.13	3600.0	823.00	829.67	-33.01	0.81	77.9
	3	1215	56.86	3600.0	758.50	764.33	-37.09	0.77	72.5
	4	1563	64.1	3600.0	830.00	843.33	-46.04	1.61	77.7
	5	1586.5	64.76	3600.0	801.00	807.67	-49.09	0.83	64.7
(40, 50)	1	826	47.97	3600.0	615.00	615.00	-25.54	0.00	58.7
	2	669.5	40.84	3600.0	600.50	605.67	-9.53	0.86	62.2
	3	894	54.12	3600.0	577.00	577.00	-35.46	0.00	43.5
	4	834.5	50.64	3600.0	638.00	639.67	-23.35	0.26	66.9
	5	1482.5	71.76	3600.0	652.50	652.50	-55.99	0.00	53.2
(40, 60)	1	1104	56.9	3600.0	736.00	741.63	-32.82	0.77	76.3
	2	1316.5	66.17	3600.0	692.50	698.50	-46.94	0.87	72.5
	3	1547.5	69.01	3600.0	688.50	690.17	-55.40	0.24	50.9
	4	1324.5	62.51	3600.0	792.00	800.67	-39.55	1.09	56.6
	5	1193.5	58.96	3600.0	752.00	759.50	-36.36	1.00	79.3
(40, 70)	1	1255.5	57.02	3600.0	812.00	819.33	-34.74	0.90	75.5
	2	1435	63.19	3600.0	832.50	839.83	-41.48	0.88	74.3
	3	1392	60.28	3600.0	823.00	826.00	-40.66	0.36	96.2
	4	1213.5	59.8	3600.0	758.50	758.50	-37.49	0.00	91.7
	5	1279	60.52	3600.0	795.00	798.33	-37.58	0.42	83.1
(40, 80)	1	1773.5	65.09	3600.0	955.00	955.00	-46.15	0.00	94.4
	2	2535.5	77.27	3600.0	908.00	911.50	-64.05	0.39	113.6
	3	2295	73.72	3600.0	948.00	953.00	-58.47	0.53	97.2
	4	2042	72.97	3600.0	882.50	894.83	-56.18	1.40	81.5
	5	1439.5	61.7	3600.0	904.40	909.47	-36.82	0.56	107.2
(50, 60)	1	1292.5	67.38	3600.0	655.00	658.67	-49.04	0.56	70.5
	2	1356.5	67.31	3600.0	682.50	698.00	-48.54	2.27	52.0
	3	1680	75.16	3600.0	684.00	687.33	-59.09	0.49	87.4
	4	1541.5	72	3600.0	663.00	663.00	-56.99	0.00	73.3
	5	1781.5	74.44	3600.0	746.50	751.67	-57.81	0.69	78.6
(50, 70)	1	2638	82.04	3600.0	746.00	755.33	-71.37	1.25	78.1
	2	1487.5	67.45	3600.0	795.00	801.83	-46.10	0.86	73.5
	3	1412	62.77	3600.0	800.00	806.83	-42.86	0.85	94.5
	4	1785	74.31	3600.0	761.00	768.50	-56.95	0.99	83.6
	5	1578	70.06	3600.0	774.50	780.83	-50.52	0.82	102.6
(50, 80)	1	2546.5	78.68	3600.0	872.00	872.00	-65.76	0.00	87.6
	2	1885.5	71.52	3600.0	861.00	866.50	-54.04	0.64	123.2
	3	1945	71.59	3600.0	926.50	928.17	-52.28	0.18	126.9
	4	3007.5	83.15	3600.0	860.50	862.83	-71.31	0.27	102.1
	5	2025	74.71	3600.0	865.50	868.50	-57.11	0.35	102.7
(50, 90)	1	2363	75.22	3600.0	962.00	964.67	-59.18	0.28	93.7
	2	2587.5	75.16	3600.0	1028.50	1035.67	-59.97	0.70	145.2
	3	3136.5	80.95	3600.0	965.00	968.50	-69.12	0.36	150.3
	4	2797.5	80.19	3600.0	963.00	964.67	-65.52	0.17	137.4
	5	3008.5	76.62	3600.0	1034.50	1048.40	-65.15	1.34	144.3

†Gap-UB: Gap between average heuristic solution and cplex incumbent after 1 CPU hour.
†Dev.: Deviation of average heuristic solution from best heuristic solution.

objective value reduction of 14.5% and an optimality gap of 0.66% on average. It is noteworthy that for similar instances, the CG scheme in Ghoniem et al. (2013) and the Benders decomposition heuristic in Solak et al. (2014) techniques exhibited an optimality gap between 4-9% within a time limit of one CPU hour.

CPLEX failed to solve any instance in Set B within one CPU hour. For all the instances in this set, the average solution constructed by our heuristic was found to significantly outperform the incumbent solution identified by CPLEX during the time limit of CPU hour. For $|S| = 35$, the heuristic consumed 61.2 CPU seconds on average and reduced the objective value of the CPLEX incumbent by 6% to 49%. In doing so, the heuristic solutions were remarkably consistent over 5 independent runs, yielding a deviation of 0.47% between the best and the average heuristic solutions obtained. Likewise, for $|S| = 40$, the heuristic converged in 76 CPU seconds on average and produced consistent solutions with a 0.53% deviation between the best and the average solutions. The average heuristic objective value reduced the incumbent objective of CPLEX by 9% to 64%. At last, for $|S| = 50$, the heuristic yielded solutions within 100 seconds on average with a 0.65% deviation between the best and the average solutions and an objective value reduction ranging between 42% and 71% over the incumbent objective value of CPLEX. In our experience, the first primal bounds identified by CPLEX during the branch-and-bound/cut process are typically poor for Set B instances and although they improve gradually over one CPU hour, the incumbent solution remains poor and is clearly outperformed by the heuristic.

3.4.2.2 Sensitivity with Respect to Objective Weights

We also assessed the robustness of the heuristic with respect to different objective weights for the routing vs. the customer assignment costs over instances in Sets A and B. We set $\lambda = 0.75$, which places greater emphasis on the routing cost, and $\lambda = 0.25$ which enforces greater importance to the customer assignment cost.

Table 3.4: Heuristic performance for Set A, $\lambda = 0.75$

		CPLEX			Heuristic			
$(\|S\|, \|K\|)$	Inst.	UB	relGap%	CPU	Best	Gap-UB% †	Gap-LB% ‡	CPU
(10, 20)	1	316.75	0	14.7	316.75	0.00	0.00	2.9
	2	317.75	0	9.6	317.75	0.00	0.00	1.9
	3	303	0	17.0	303	0.00	0.00	2.0
	4	365.5	0	22.4	365.5	0.00	0.00	2.7
	5	372.5	0	220.2	372.5	0.00	0.00	2.7
(10, 30)	1	429.5	0	451.1	429.5	0.00	0.00	6.0
	2	439	0	169.6	439	0.00	0.00	1.9
	3	430.25	0	1760.3	430.25	0.00	0.00	2.3
	4	471.25	0	3525.5	471.25	0.00	0.00	4.8
	5	505.5	17.03	3600.0	509.75	0.84	1.15	3.8
(10, 40)	1	594.5	15.62	3600.0	591.25	-0.33	0.43	9.6
	2	565.5	4.8	3600.0	565.5	1.54	1.67	3.9
	3	625	13.86	3600.0	625	0.00	0.01	4.7
	4	583.5	14.1	3600.0	579.75	-0.64	0.00	6.3
	5	570.25	19.53	3600.0	568	-0.39	0.00	2.4
(10, 50)	1	872.75	33.26	3600.0	826.25	-5.16	0.40	14.3
	2	741.5	10.86	3600.0	712.25	-3.94	0.00	10.2
	3	702.5	33.79	3600.0	649	-7.62	0.00	6.6
	4	813	36.1	3600.0	696.5	-12.63	1.98	8.1
	5	767	30.17	3600.0	692.25	-9.75	0.00	8.5
(25, 20)	1	288.5	33.49	3600.0	277.25	-3.33	0.60	5.7
	2	269.5	31.35	3600.0	269.5	0.00	0.00	4.8
	3	316.25	34.4	3600.0	307.5	-2.77	0.00	7.2
	4	293	38.05	3600.0	289	-1.37	0.00	8.5
	5	286.75	29.43	3600.0	286.75	0.00	0.00	4.5
(25, 30)	1	401	32.14	3600.0	390	-2.74	0.00	13.2
	2	444	48.29	3600.0	418.75	-5.69	0.14	10.1
	3	409.5	32.42	3600.0	403	-1.59	0.49	15.2
	4	523.75	49.94	3600.0	440.25	-15.94	0.26	11.6
	5	532.75	58.64	3600.0	431.25	-18.98	0.09	9.8
(25, 40)	1	554.25	43.96	3600.0	486.25	-12.27	1.60	17.2
	2	638	54.93	3600.0	530	-16.93	0.09	22.4
	3	616	55.03	3600.0	526.5	-14.20	0.88	30.0
	4	581.5	42.54	3600.0	516.25	-11.22	0.00	8.2
	5	591.75	49.41	3600.0	493.5	-16.60	0.00	14.4
(25, 50)	1	786.25	56.95	3600.0	653.25	-16.92	0.40	16.3
	2	958.75	65.09	3600.0	629	-34.34	0.49	30.6
	3	776.25	58.77	3600.0	599.5	-22.77	0.58	39.0
	4	797.25	60.56	3600.0	637.75	-20.01	0.43	32.4
	5	1008.25	67.7	3600.0	648.25	-35.71	0.50	34.1

†Gap-UB: Gap between average heuristic solution and cplex incumbent after 1 CPU hour.
‡Gap-LB: Gap between average heuristic solution and CG-based LB.

Table 3.5: Heuristic performance for Set B, $\lambda = 0.75$

		CPLEX			Heuristic Performance				
$(\|S\|, \|K\|)$	Inst.	UB	relGap%	CPU	Best	Avg	Gap-UB% †	Dev.% ‡	CPU
(35, 40)	1	564.25	52.57	3600.0	467.25	469.80	-16.74	0.55	41.5
	2	619.5	55.29	3600.0	488	491.35	-20.69	0.69	51.1
	3	572.25	56.21	3600.0	470	470.00	-17.87	0.00	47.0
	4	718.5	65.08	3600.0	496.5	496.50	-30.90	0.00	41.1
	5	770.25	66.74	3600.0	491.5	494.10	-35.85	0.53	49.2
(35, 50)	1	739.25	59.29	3600.0	510	513.40	-30.55	0.67	40.0
	2	661	54.55	3600.0	567	570.65	-13.67	0.64	42.5
	3	830.5	67.77	3600.0	593.75	606.10	-27.02	2.08	61.7
	4	1033.75	75.03	3600.0	589.75	594.10	-42.53	0.74	47.5
	5	1157.5	70.69	3600.0	627.5	630.70	-45.51	0.51	44.0
(35, 60)	1	1158.75	73.51	3600.0	647.5	652.60	-43.68	0.79	62.3
	2	1154.5	69.99	3600.0	647.25	654.00	-43.35	1.04	64.1
	3	822.5	58.52	3600.0	614.5	621.25	-24.47	1.10	51.0
	4	1007.75	67.39	3600.0	673	678.40	-32.68	0.80	52.8
	5	1016	66.42	3600.0	687	690.55	-32.03	0.52	47.2
(35, 70)	1	1244.5	67.89	3600.0	752	758.20	-39.08	0.82	70.1
	2	1085.25	64.7	3600.0	723.75	729.70	-32.76	0.82	59.9
	3	983.5	64.81	3600.0	745.5	750.25	-23.72	0.64	100.8
	4	1016	62.15	3600.0	750	754.40	-25.75	0.59	59.2
	5	1056.5	63.37	3600.0	728	732.55	-30.66	0.62	53.8
(40, 50)	1	869.75	67.99	3600.0	510	510.33	-41.32	0.07	41.5
	2	839.75	68.36	3600.0	566	566.00	-32.60	0.00	53.7
	3	1021.25	73.59	3600.0	499.25	499.25	-51.11	0.00	47.5
	4	837.5	68.06	3600.0	570	573.67	-31.50	0.64	48.5
	5	1141.75	76.79	3600.0	603	604.75	-47.03	0.29	49.7
(40, 60)	1	1004.75	67.61	3600.0	624.25	627.58	-37.54	0.53	69.5
	2	1123	71.69	3600.0	677	677.00	-39.72	0.00	61.0
	3	1177.25	72.63	3600.0	631	631.00	-46.40	0.00	74.2
	4	840.75	61.84	3600.0	683.5	685.83	-18.43	0.34	53.1
	5	1108.25	71.36	3600.0	691.5	692.33	-37.53	0.12	62.2
(40, 70)	1	1218.25	69.41	3600.0	752.75	754.75	-38.05	0.27	81.5
	2	1407	74.93	3600.0	723.5	725.92	-48.41	0.33	59.8
	3	1284.25	71.5	3600.0	717	726.33	-43.44	1.30	80.8
	4	1258.25	73.73	3600.0	735.5	747.25	-40.61	1.60	88.4
	5	1113.75	68.73	3600.0	749.5	750.92	-32.58	0.19	103.1
(40, 80)	1	1313.5	68.83	3600.0	834.5	843.42	-35.79	1.07	130.6
	2	1378	73.5	3600.0	844.5	847.83	-38.47	0.39	111.9
	3	1869.75	79.31	3600.0	880.75	883.00	-52.77	0.26	118.0
	4	1258.5	70.7	3600.0	853.25	854.58	-32.10	0.16	118.5
	5	1309.5	70.83	3600.0	859.5	862.67	-34.12	0.37	115.3
(50, 60)	1	1054	71.77	3600.0	631.25	635.00	-39.75	0.59	55.2
	2	1195.75	75.48	3600.0	631.5	631.50	-47.19	0.00	58.6
	3	1265.5	78.1	3600.0	643.25	652.00	-48.48	1.36	55.1
	4	1329	78.35	3600.0	615	616.25	-53.63	0.20	72.0
	5	1274.25	76.46	3600.0	684	689.25	-45.91	0.77	59.4
(50, 70)	1	1503.5	77.63	3600.0	727.25	730.67	-51.40	0.47	66.4
	2	1312.75	75.09	3600.0	720	730.08	-44.39	1.40	66.3
	3	1767.25	79.92	3600.0	728.25	730.25	-58.68	0.27	61.4
	4	1292.25	76.48	3600.0	735.5	742.25	-42.56	0.92	59.6
	5	1448.25	78.18	3600.0	766.25	768.25	-46.95	0.26	64.8
(50, 80)	1	1681.25	78.77	3600.0	833.25	836.83	-50.23	0.43	63.9
	2	1274.25	71.67	3600.0	826.5	829.00	-34.94	0.30	92.5
	3	1618.25	77.82	3600.0	874.5	879.33	-45.66	0.55	123.4
	4	1549.75	78.87	3600.0	839.5	840.58	-45.76	0.13	80.2
	5	1530.5	79.11	3600.0	840.75	844.08	-44.85	0.40	103.3
(50, 90)	1	1947.75	82.08	3600.0	907	907.00	-53.43	0.00	122.4
	2	2081.75	82.7	3600.0	944.5	946.17	-54.55	0.18	160.0
	3	1869.5	79.93	3600.0	949.5	950.83	-49.14	0.14	198.9
	4	1800	80.06	3600.0	933.5	937.58	-47.91	0.44	104.4
	5	1827	74.31	3600.0	961	970.33	-46.89	0.97	118.6

†Gap-UB: Gap between average heuristic solution and cplex incumbent after 1 CPU hour.
†Dev.: Deviation of average heuristic solution from best heuristic solution.

Table 3.6: Heuristic performance for Set A, $\lambda = 0.25$

		CPLEX			Heuristic			
$(\lvert S\rvert, \lvert K\rvert)$	Inst.	UB	relGap%	CPU	Best	Gap-UB% †	Gap-LB% ‡	CPU
(10, 20)	1	462.75	0	7.1	462.75	0.00	0.05	3.6
	2	448.25	0	1.2	448.25	0.00	0.00	1.9
	3	381.25	0	102.5	381.25	0.00	0.36	1.9
	4	493.25	0	67.7	493.25	0.00	0.00	1.6
	5	376.5	0	212.7	376.5	0.00	0.00	2.1
(10, 30)	1	586.5	0	297.8	586.5	0.00	0.00	2.7
	2	630	0	8.1	630	0.00	0.00	2.1
	3	503.75	0.15	3600.0	504.5	0.15	0.15	2.7
	4	750.75	1.58	3600.0	750.75	0.17	1.20	9.6
	5	576.5	5.97	3600.0	576.5	0.00	0.04	7.3
(10, 40)	1	850.25	3.56	3600.0	850.25	0.00	0.00	7.0
	2	856.75	0	633.3	856.75	0.00	0.00	3.1
	3	923.75	5.07	3600.0	923.75	0.00	0.12	5.2
	4	730.25	5.64	3600.0	731.25	0.14	1.26	6.5
	5	855.5	5.91	3600.0	855.5	0.00	0.00	2.3
(10, 50)	1	1177.25	8.68	3600.0	1134	-3.67	0.01	3.2
	2	1077.75	0	1866.1	1077.75	0.00	0.00	7.5
	3	876.25	7.89	3600.0	856.25	-2.28	0.41	2.6
	4	1055	5.54	3600.0	1048	-0.66	0.44	7.6
	5	999.25	2.55	3600.0	995.5	-0.38	0.00	5.7
(25, 20)	1	328.25	4.41	3600.0	328.25	0.00	0.00	11.8
	2	306.25	5.42	3600.0	306.25	0.03	0.03	15.3
	3	366.5	9.75	3600.0	366.5	0.00	0.00	10.7
	4	314.75	6.58	3600.0	314.75	0.41	0.41	16.2
	5	310	8.34	3600.0	310	0.00	0.00	6.5
(25, 30)	1	427.75	9.77	3600.0	421.75	-0.95	0.92	22.6
	2	413.75	10.19	3600.0	413.75	0.00	0.00	10.2
	3	473.75	9.27	3600.0	470.75	-0.28	0.35	20.7
	4	523	10.76	3600.0	518.25	-0.91	0.16	13.8
	5	448.5	20.61	3600.0	435.75	-2.84	0.00	12.5
(25, 40)	1	591.25	21.58	3600.0	513.25	-13.19	0.00	24.2
	2	606	19.9	3600.0	567.25	-6.39	0.00	19.5
	3	634	28.32	3600.0	547	-12.37	3.38	38.7
	4	705.75	17.53	3600.0	643.75	-8.78	0.64	33.5
	5	624	15.61	3600.0	586.75	-5.68	0.51	33.2
(25, 50)	1	794.25	23.3	3600.0	715.75	-9.88	0.00	33.9
	2	880	22.78	3600.0	779.5	-11.24	0.36	64.5
	3	766.5	28.88	3600.0	633.5	-17.35	0.34	40.9
	4	791.25	28.34	3600.0	682.75	-13.71	0.51	36.4
	5	955.75	43.49	3600.0	670.25	-29.66	1.63	44.7

†Gap-UB: Gap between average heuristic solution and cplex incumbent after 1 CPU hour.
‡Gap-LB: Gap between average heuristic solution and CG-based LB.

Table 3.7: Heuristic performance for Set B, $\lambda = 0.25$

		CPLEX			Heuristic Performance				
$(\|S\|, \|K\|)$	Inst.	UB	relGap%	CPU	Best	Average	Gap-UB% †	Dev.% ‡	CPU
(35, 40)	1	519.25	15.28	3600	498	500.17	-3.68	0.44	43.7
	2	564.5	21.08	3600	507.5	508.00	-10.01	0.10	43.4
	3	567.75	27.8	3600	488.25	488.58	-13.94	0.07	50.5
	4	617	25.2	3600	534.5	534.50	-13.37	0.00	45.1
	5	655.5	29.19	3600	535.5	535.50	-18.31	0.00	43.5
(35, 50)	1	717.25	28.75	3600	598.25	599.42	-16.43	0.20	50.4
	2	662	21.93	3600	592	592.00	-10.57	0.00	40.9
	3	1119.75	50.48	3600	645.5	649.33	-42.01	0.59	52.9
	4	621.25	26.32	3600	549	554.08	-10.81	0.93	52.1
	5	664.75	36.31	3600	534.25	535.08	-19.51	0.16	43.8
(35, 60)	1	713.5	25.77	3600	642.75	642.75	-9.92	0.00	48.8
	2	852	28.3	3600	706.75	710.25	-16.64	0.50	55.2
	3	832.25	33.45	3600	640	640.08	-23.09	0.01	46.9
	4	829.25	31.46	3600	691.5	693.33	-16.39	0.27	62.4
	5	1404	53.62	3600	780	789.67	-43.76	1.24	63.9
(35, 70)	1	935.25	25.13	3600	826.5	826.50	-11.63	0.00	59.2
	2	1921.75	66.73	3600	776	777.08	-59.56	0.14	72.0
	3	1121	46.39	3600	718.75	718.75	-35.88	0.00	46.7
	4	1959.75	66.1	3600	793.25	797.92	-59.28	0.59	55.1
	5	1732.25	63.24	3600	746	746.00	-56.93	0.00	48.7
(40, 50)	1	679.75	28.16	3600	568.75	569.25	-16.26	0.09	69.1
	2	649.5	30.67	3600	530	532.33	-18.04	0.44	48.4
	3	653.25	27.09	3600	550.25	550.42	-15.74	0.03	49.7
	4	655.25	27.02	3600	579	579.83	-11.51	0.14	50.8
	5	828.75	37.86	3600	617.5	619.58	-25.24	0.34	52.0
(40, 60)	1	1615.5	65.09	3600	679.25	683.08	-57.72	0.56	77.1
	2	813.5	34.89	3600	631	635.00	-21.94	0.63	59.0
	3	800.75	32.02	3600	637.75	641.92	-19.84	0.65	51.8
	4	1841.75	66.83	3600	738.25	740.92	-59.77	0.36	65.3
	5	2077.5	71.89	3600	718.25	718.50	-65.42	0.03	67.1
(40, 70)	1	1665.25	63.83	3600	735.5	738.67	-55.64	0.43	62.3
	2	1636.5	62.57	3600	757.5	758.50	-53.65	0.13	63.8
	3	1435.25	54.6	3600	781.25	782.83	-45.46	0.20	68.1
	4	1449.25	60.06	3600	713.25	714.25	-50.72	0.14	63.3
	5	1302	56.1	3600	714	719.50	-44.74	0.77	50.8
(40, 80)	1	2229.25	67.38	3600	875.25	882.92	-60.39	0.88	87.0
	2	2043.75	64.25	3600	856.75	861.25	-57.86	0.53	105.9
	3	1872.25	60.34	3600	905.25	907.00	-51.56	0.19	88.2
	4	2262.75	71.02	3600	810	819.33	-63.79	1.15	79.9
	5	1417.25	55.4	3600	799	799.83	-43.56	0.10	77.3
(50, 60)	1	1189.5	61.98	3600	556.75	557.42	-53.14	0.12	83.1
	2	1795.25	72.74	3600	623.5	623.50	-65.27	0.00	78.2
	3	842	40.31	3600	617	617.00	-26.72	0.00	75.6
	4	725.75	30.83	3600	600.5	600.67	-17.24	0.03	68.3
	5	2535	78.71	3600	686.5	687.33	-72.89	0.12	83.8
(50, 70)	1	2017.75	74.46	3600	642.5	649.17	-67.83	1.04	118.3
	2	2496	77.49	3600	711.25	717.25	-71.26	0.84	70.7
	3	2465.25	74.69	3600	763.25	764.17	-69.00	0.12	82.0
	4	2573	78.39	3600	692.5	697.00	-72.91	0.65	147.3
	5	2152.25	74.12	3600	699.25	701.67	-67.40	0.35	87.4
(50, 80)	1	2756.25	77.09	3600	789	792.17	-71.26	0.40	92.1
	2	1835	66.41	3600	788	790.92	-56.90	0.37	90.7
	3	2877	77.24	3600	835.25	840.08	-70.80	0.58	108.1
	4	3129	80.28	3600	777	778.00	-75.14	0.13	80.4
	5	2959.5	79.33	3600	770.25	773.83	-73.85	0.47	68.5
(50, 90)	1	2245.5	69.05	3600	876.75	881.00	-60.77	0.48	105.1
	2	3641	78.97	3600	952	958.17	-73.68	0.65	94.7
	3	3143.5	77.51	3600	878.5	878.50	-72.05	0.00	91.1
	4	3676.75	81.39	3600	871.5	876.67	-76.16	0.59	99.4
	5	3466.5	75	3600	960.25	964.08	-72.19	0.40	89.2

†Gap-UB: Gap between average heuristic solution and cplex incumbent after 1 CPU hour.
†Dev.: Deviation of average heuristic solution from best heuristic solution.

For $\lambda = 0.75$, the results reported in Table 3.4 indicate that CPLEX successfully solved 9/40 instances in Set A to optimality within one CPU hour. In contrast, the average CPU time of the heuristic for $|S| = 10$ and $|S| = 25$ is 5.27 and 16.7 seconds, respectively. For $|S| = 10$, the heuristic solution was at least as good as the incumbent solution of CPLEX in 18/20 instances and strictly outperformed CPLEX in 8/20 instances with a cost reduction of 5% on average. For $|S| = 25$, the heuristic solution was at least as good as the CPLEX solution in all 20 instances and improved upon the CPLEX solution in 18/20 instances with a cost reduction of 14.1% on average. The heuristic results also substantially outperform the CG results in Ghoniem et al. (2013) which achieved an optimality gap between 5-6% on average for comparable instances with $\lambda = 0.75$.

According to Table 3.5, for Set B and $\lambda = 0.75$, CPLEX failed to solve all instances within one hour and produced incumbent solutions that are significantly outperformed by our heuristic solutions. For $|S| = 35$, the heuristic terminated in 54.3 CPU seconds on average and yielded a deviation of 0.71% between the best and average heuristic solution. Similar observations can be made for $|S| = 40$ and $|S| = 50$. The objective value reduction achieved by the heuristic over the incumbent by CPLEX ranged from 13% to 54%.

Tables 3.6 and 3.7 report our results for Set A and B respectively with $\lambda = 0.25$. Whereas CPLEX solved 9/20 instances to optimality within the specified time limit, the proposed heuristic provided overall higher quality solutions in significantly shorter CPU times. For example, with $|S| = 25$, the heuristic solution improved upon the CPLEX solution in 14/20 instances with a cost reduction of 9.52% on average. For Set B, CPLEX could not solve any instances to optimality within one CPU hour. For these instances, the average heuristic solution is significantly better than the primal bounds identified by CPLEX. The heuristic converged in 51.3, 66.8 and 90.7 CPU seconds for $|S| = 35, 40,$ and 50, with an accompanying deviation of 0.26%, 0.39%, and 0.37% between the best and average heuristic solutions, for instances with $|S| = 35, 40$, and 50, respectively. The objective value reduction achieved by the heuristic over the CPLEX incumbent ranged from 3% to a substantial 76% over Set B instances.

In summary, for instances in Set A, the heuristic is robust with respect to different weights for routing and customer assignment costs. For instances in Set B, no conclusion can be made, because no reliable primal or dual bounds are available for assessing the performance of the heuristic. However, the heuristic produced consistent solutions over 5 independent runs, exhibiting deviation below 1% between the best and the average heuristic solutions, for $\lambda = 0.25$, 0.5, and 0.75. However, it seems that this deviation is noticeably lower for $\lambda = 0.25$ (greater emphasis on customer assignment cost) in which case the heuristic escapes local optima more effectively. This will be noted in next section as well when we discuss the impact of perturbation mechanisms on the quality of the heuristic solution.

3.4.2.3 Comparative Analysis of Different Algorithmic Schemes

This section investigates the impact of key algorithmic components of Algorithm 1 on the performance of the proposed heuristic. To this end, we examine the performance of the following three variants of Algorithm 1: (i) Variant A in which Procedure CVRP(π) (line 10 of Algorithm 1) is eliminated; (ii) Variant B in which Procedure SITEREMOVAL(π) (line 12 of Algorithm 1) is eliminated; and (iii) Variant C in which Procedure PERTURB(π) (line 20 of Algorithm 1) is eliminated. Note that we did not consider elimination of Procedure GAP(π), because it is necessary for assigning customers to delivery sites in different parts of the heuristic. All the variants are tested for all the instances with $|S| = 25$ and $|S| = 50$. For each instance, the gap between the solution of each variant and the base heuristic solution is computed. The average gap value over the five instances for each $(|S|, |K|)$ combination is reported in Table 3.8. Our main observations are summarized next:

- Variant A: For all λ values, Procedure $\textsc{Cvrp}(\pi)$ has a noticeable impact on the solution quality of heuristic. However, as we increase the weight of routing cost, the impact of Procedure $\textsc{Cvrp}(\pi)$ decreases. For example, for $|S| = 50$ and for $\lambda = 0.25$, the average gap between solution of Variant A and base heuristic is 2.99%. As we increase λ, this gap reduces to 1.33% and 0.73% for $\lambda = .5$ and 0.75, respectively. In fact, for larger λ values, greater emphasis is placed on the routing cost which, in turn, limits the use of delivery sites. On the other hand, giving more importance to customer assignment cost (by reducing λ) encourages the use of a greater number of delivery sites in order to reduce the assignment cost. Consequently, when a larger number of delivery sites is used in the solution, there is a greater need for adjusting routing decisions and using the local search $\textsc{Cvrp}(\pi)$ has a more noticeable impact.

- Variant B: The exclusion of Procedure $\textsc{SiteRemoval}(\pi)$ has worsened the performance of heuristic for all λ values. However, its impact is more significant as we increase the value of λ. In fact, as we increase the value of λ, the number of delivery sites in optimal or near-optimal solutions of a given instance decreases and, hence, the ability to remove inferior delivery sites becomes more important.

- Variant C: Procedure $\textsc{Perturb}(\pi)$ also plays an important role in the quality of solutions. Its impact increases as more weight is assigned to the routing cost (increase in the λ value). As pointed out in Section 5.2.2, the heuristic solutions exhibit less variation for lower λ values. When $\lambda = 0.25$ the heuristic has less trouble escaping local optima.

3.5. Conclusion

This chapter proposes a multi-start heuristic for a vehicle routing with demand allocation problem that arises in the distribution of pallets of goods by food banks.

Table 3.8: Comparative Analysis of Different Algorithmic Schemes

	$\lambda = .25$			$\lambda = 0.5$			$\lambda = 0.75$		
$(\lvert S\rvert, \lvert K\rvert)$	Variant A	Variant B	Variant C	Variant A	Variant B	Variant C	Variant A	Variant B	Variant
(25, 20)	0.91	0.69	1.23	0.78	4.24	2.25	0.41	32.58	2.70
(25, 30)	1.03	0.27	0.35	0.81	3.15	0.77	0.78	23.02	4.17
(25, 40)	1.89	0.55	0.85	0.69	4.21	0.95	0.54	23.94	1.59
(25, 50)	2.41	0.32	0.67	0.86	2.63	0.85	0.31	17.98	0.61
(50, 60)	2.53	0.50	0.69	0.58	2.13	0.89	0.48	20.21	0.88
(50, 70)	2.82	0.34	0.74	1.22	2.53	1.03	0.60	17.69	1.40
(50, 80)	2.83	0.29	0.45	1.74	2.50	1.14	0.70	16.28	2.08
(50, 90)	3.78	0.35	0.60	1.78	2.02	1.40	1.12	16.68	2.61

The objective is to minimize an average weighted vehicle routing and customer travel costs. We generated two sets of instances, Set A and Set B, comprising 100 instances. For each instance in our computational study, three different objective weights were considered, placing equal or greater emphasis on either the vehicle routing cost or the customer travel cost. In no more than 70 CPU seconds, the proposed multi-start heuristic consistently yielded solutions within 0.5% of optimality for instances of Set A in our testbed for which a lower bound was computed by column generation techniques. For larger instances in Set B, although the heuristic could not be compared to a lower bound, heuristic solutions substantially outperformed the best incumbent solution identified by CPLEX within one CPU hour. These encouraging results compare favorably and outperform the column generation approach in Ghoniem et al. (2013) and the Benders decomposition heuristic in Solak et al. (2014) which required one CPU hour for comparable instances and resulted in significantly higher optimality gaps (between 4-9%).

The superior performance of the proposed heuristic is grounded in its ability to diversify the search by altering the initial solution at each start of the algorithm and to intensify the neighborhood exploration using a series of local improvement schemes. Our computational findings demonstrate the efficacy of optimization-based heuristics that involve fast neighborhood explorations in order to address this type of vehicle routing-allocation problems. Such approaches provide a computationally attractive alternative to classical column generation and Benders decomposition techniques that exhibit slow convergence patterns. We recommend for future research the investigation of a multi-period variant of this vehicle routing with demand allocation problem with fixed delivery points to be determined for time-varying customer demands.

CHAPTER 4

A BRANCH-AND-PRICE ALGORITHM FOR A VEHICLE ROUTING-ALLOCATION PROBLEM

In this chapter, we investigate the vehicle routing with demand allocation problem where the decision-maker jointly optimizes the location of delivery sites, the assignment of customers to (preferably convenient) delivery sites, and the routing of vehicles operated from a central depot to serve customers at their designated sites. We propose an effective branch-and-price (B&P) algorithm that is demonstrated to greatly outperform the use of commercial branch-and-bound/cut solvers such as CPLEX. Central to the efficacy of the proposed B&P algorithm is the development of a specialized dynamic programming procedure that extends works on elementary shortest path problems with resource constraints in order to solve the more complex column generation pricing subproblem. Our computational study demonstrates the efficacy of the proposed approach using a set of 60 problem instances. Moreover, the proposed methodology has the merit of providing optimal solutions in run times that are significantly shorter than those reported for decomposition-based heuristics in the literature.

4.1. Introduction and Motivation

This chapter addresses the Vehicle Routing with Demand Allocation Problem (VRDAP) that arises in food bank distribution planning, as investigated in Ghoniem et al. (2013) and Solak et al. (2014). The decision-maker seeks to jointly optimize the selection of delivery sites from a set of candidate locations, to assign geographically

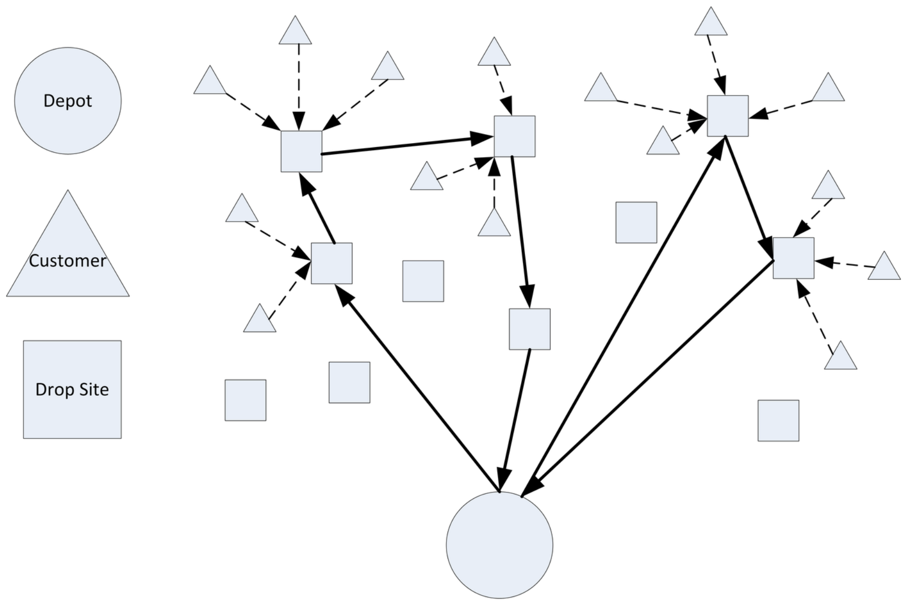

Figure 4.1: Network structure of the VRAP variant under investigation

dispersed customers to (preferably convenient) delivery sites, and to route a fleet of vehicles from a central depot owned by a food bank in order to deliver goods or services to customers at their designated sites. In this context, delivery sites are typically parking lots owned by large retailers or religious establishments that consent to their use and their access is restricted to at most one delivery tour. As a consequence, vehicle routes are disjoint and delivery sites are implicitly capacitated, because their allocated customer demand cannot exceed the capacity of delivery vehicles. As illustrated in Figure 4.1, both customers and vehicles travel to meet at delivery sites and it is desirable to minimize the total distance traveled by vehicles and customers. This integrated problem is NP-hard and poses significant computational challenges. In fact, when customers are feasibly assigned to pre-selected delivery sites, the problem reduces to the classical capacitated VRP.

This chapter contributes to enhancing the tractability of this problem by developing a specialized branch-and-price (B&P) algorithm that substantially and consistently outperforms CPLEX on all tested problem instances. Second, it provides optimal solutions in significantly shorter run times than a column generation heuristic (Ghoniem et al. 2013) and a Benders decomposition heuristic (Solak et al. 2014) which exhibit 4-9% optimality gaps over instances having up to 25 delivery sites and 50 customers. Third, from a heuristic point of view, the root-node solutions of the proposed B&P algorithm exhibit excellent optimality gaps (below 1% on average) in run times that are much shorter than those for the aforementioned decomposition-based heuristics in the literature. From a methodological point of view, the column generation pricing subproblem is solved with a specialized dynamic programming (DP) algorithm that extends works on elementary shortest path problems with resource constraints (ESPPRC) in order to solve the pricing subproblem. In fact, the pricing subproblem involves a single-vehicle VRDAP with prize-collecting considerations and jointly optimizes the selection of delivery sites, the assignment of customers to delivery sites, and the routing of delivery vehicles. To the best of our knowledge, this is the first DP algorithm that extends works on the ESPPRC to account for these additional decisions. This chapter also explores heuristic variants of the DP algorithm that greatly contribute to accelerating the construction of new attractive columns, whereas the more computationally involved exact DP algorithm is invoked with parsimony, mostly to achieve LP optimality at a given node of the B&P tree.

The remainder of this chapter is organized as follows. Section 4.2 discusses the overall design of the branch-and-price algorithm by presenting a set partitioning formulation of the problem, a mixed-integer programming model for the pricing subproblem, and the chosen branching strategy. We then elaborate in Section 4.3 on solving the pricing subproblem using exact and heuristic DP procedures. Section 4.4 discusses our computational results and demonstrates the efficacy of the proposed methodology. Section 4.5 concludes the chapter with a summary of our findings and recommends directions for future research.

4.2. Branch-and-Price Algorithm

We present a branch-and-price (BP) algorithm for the VRDAP. We first formulate the VRDAP as a set partitioning model which serves as a master program (MP) in Section 4.2.1. The master program involves an exponential number of columns, each associated with a feasible route, which could be onerous to generate and solve using commercial solvers. Instead, a BP algorithm (Algorithm 1) is developed to solve the MP to optimality. To this end, the underlying LP relaxation of the MP is solved by column generation technique (Algorithm 2) and branching is performed on fractional LP solutions. Next, we briefly discuss the general framework of the proposed BP algorithm.

Algorithm 1 starts by generating an initial solution using a heuristic proposed in section 4.2.2. The constructed solution is considered as the current best solution (upper bound) (line 3). The LP relaxation of the MP is solved by CG, yielding a lower bound, at the root node of the BP tree (line 5). If the gap between the lower and upper bounds equals zero, the problem has been solved to optimality and the BP algorithm stops. Otherwise, branching is performed on fractional solutions on routing or assignment arcs using a best-first strategy as described in section 4.2.4.

In the CG procedure (Algorithm 2), the continuous restricted master program (RMP) is solved using CPLEX; the associated dual values are then used in the pricing subproblem presented in section 4.2.3, for which we have devised exact and heuristic dynamic programming (DP) algorithms. For computational efficiency, the CG approach first invokes several heuristic versions of the DP algorithm, as detailed in section 4.3.3, and adds a set of routes with the most negative reduced cost to the RMP. If all the heuristic DPs fail to construct a column having a negative reduced cost, then the exact DP of section 4.3.1 is solved instead as a last resort. If the exact DP also fails to find a column having a negative reduced cost, then the CG terminates and the continuous RMP is solved to optimality.

Algorithm 4 BP Algorithm

```
1:  Comment: Let $\Delta$ be the list of all the open nodes of BP
2:  Comment: Let $\delta_{lb}$ be the solution of LP relaxation of node $\delta$ computed using CG.
3:  $Best \leftarrow$ The constructed initial solution
4:  Construct root node $\bar{\delta}$ and initialize its RMP using the routes of $Best$
5:  Compute $\bar{\delta}_{lb}$ using CG procedure (Algorithm 2)
6:  $\Delta \leftarrow \{\bar{\delta}\}$
7:  while $\Delta \neq \varnothing$ do
8:     $\delta \leftarrow$ The node in $\Delta$ with the minimum lower bound
9:     $\Delta \leftarrow \Delta \setminus \delta$
10:    if $\delta_{lb}$ is better than $Best$ then
11:       if $\delta_{lb}$ is feasible (integral) then
12:          $Best \leftarrow \delta_{lb}$
13:       else
14:          Construct $\delta^1$ and $\delta^2$; the nodes obtained from branching on $\delta$
15:          Compute $\delta^1_{lb}$ and $\delta^2_{lb}$ using CG procedure (Algorithm 2)
16:          $\Delta \leftarrow \Delta \cup \{\delta^1, \delta^2\}$
17:       end if
18:    end if
19: end while
```

Algorithm 5 Column Generation (CG)

```
1:  while true do
2:     Solve the LP relaxation of RMP using CPLEX
3:     Try to find promising columns using heuristic DPs of section 4.3.3
4:     if A heuristic found columns with negative reduced cost then
5:        Add up to 20 of the columns with most negative cost to RMP
6:     else
7:        Call Exact DP of section 4.3.1
8:        if Columns with negative reduced cost were found then
9:           Add up to 20 of the columns with most negative cost to RMP
10:       else
11:          Solution of RMP is the optimal solution of LP relaxation of MP
12:          Terminate the algorithm
13:       end if
14:    end if
15: end while
```

4.2.1 Problem Statement and Set Partitioning Formulation

We consider a VRDAP involving a central depot, a set of delivery sites, S, and a set of customers, K. We denote by V the maximum number of vehicles/tours of capacity Q that can be used for delivery purposes. Any tour commences at the central depot, sequentially visits a subset of delivery sites in order to supply goods to customers, and ends at the central depot. Any delivery site is accessed by at most one tour and, therefore, all tours are assumed to be disjoint. Each customer $k \in K$ has a demand d_k that is allocated to some delivery site that lies on a tour. Let $N = S \cup \{0\}$ be the set of delivery sites augmented with node 0 that represents the central depot. Also, let $E = \{(i,j), i,j \in N, i \neq j\}$ and $E^{'} = \{(k,j), k \in K, j \in S\}$ be the set of routing and assignment arcs, respectively. We define c_{ij} as the cost of arc $(i,j) \in E$ and f_{kj} as the cost of assigning customer k to site j, where $(k,j) \in E^{'}$.

The VRDAP can be modeled as a 0-1 MIP (as in the Appendix A) or equivalently as a set partitioning model with packing constraints that lends itself to column generation (CG) approaches. We denote by H the set of all columns that correspond to feasible tours where the total customer demand assigned to any delivery site does not exceed the tour capacity. For each such tour $h \in H$, let P^h and Q^h be 0-1 vectors that respectively list customers and delivery sites that are associated with tour h and let ρ_h be the total vehicle routing and customer travel cost incurred in tour h. Using a binary variable z_h that indicates whether tour h is selected or not, the following master program (MP) is formulated:

$$\textbf{MP:}\ \text{Minimize} \sum_{h \in H} \rho_h z_h \tag{4.1a}$$

$$\text{subject to} \sum_{h \in H} P_k^h z_h = 1 \qquad \forall k = 1, \ldots, |K| \tag{4.1b}$$

$$\sum_{h \in H} Q_i^h z_h \leq 1 \qquad \forall i = 1, \ldots, |S| \tag{4.1c}$$

$$\sum_{h \in H} z_h = |V| \tag{4.1d}$$

$$z \text{ binary.} \tag{4.1e}$$

The objective function (4.1a) minimizes the total vehicle and customer travel cost. Constraint (4.1b) ensures that every customer is served by exactly one vehicle tour, thereby achieving a partitioning scheme for customers. Constraint (4.1c) guarantees that any delivery site is visited by at most one vehicle tour. Constraint (4.1d) imposes that $|V|$ tours are used, where $|V|$ is the maximum number of vehicle tours possible. We state this constraint as an equality to avoid branching on the number of vehicles used and allow the inclusion of dummy, vacuous tours from the central depot to itself with a zero cost, should the actual number of tours used be below $|V|$.

4.2.2 Initial Solution

This section briefly describes a heuristic that is used for obtaining an initial solution for the VRDAP. The heuristic solves a generalized assignment problem to find a minimum cost assignment of customers to delivery sites, while ensuring that the aggregate demand assigned to each site does not exceed the vehicle capacity. Next, routes are constructed using a nearest neighbor procedure. Let S^+ be the set of all delivery sites having at least one customer assigned to them. Starting from the depot, the vehicle travels to the nearest delivery site in S^+ such that its inclusion in the route does not violate the vehicle capacity. The newly visited delivery site is then removed from S^+. This procedure iterates until no delivery site can be feasibly added to the route or $S^+ = \varnothing$. This procedure fails to generate a feasible solution if all vehicles have been used, but unassigned delivery sites remain. In this case, ignoring vehicle capacity considerations, the remaining delivery sites are inserted in a route in a manner that results in a minimum insertion cost. All customers are reassigned to the routes by solving a second generalized assignment problem where the assignment cost of each customer k to a route r is the distance between k and its nearest delivery site in r. If the problem instance at hand is feasible, then this latter step must yield a feasible solution, which is then improved using CVRP local search procedures.

4.2.3 Pricing Subproblem

The CG pricing subproblem constructs a feasible VRDAP route with a minimum reduced cost, using the dual values obtained from the LP solution of the RMP. If the constructed route has negative reduced cost, its corresponding column is added to the RMP. Otherwise, the LP procedure terminates with an optimal solution to the continuous relaxation of the MP. To introduce the CG pricing subproblem, consider the following notation:

- $x_i \in \{0,1\}$: $x_i = 1$ if and only if site i is selected in the constructed column, $\forall i \in S$.
- $y_k \in \{0,1\}$: $y_k = 1$ if and only if customer k is selected in the constructed column, $\forall k \in K$.
- $e_{ij} \in \{0,1\}$: $e_{ij} = 1$ if arc (i,j) is included in the constructed route, $\forall (i,j) \in E$.
- $s_{ik} \in \{0,1\}$: $s_{ik} = 1$ if customer k is assigned to node i, $\forall i \in S, k \in K$.
- q_i : Total deliveries made upon serving site i, $\forall i \in S$.
- μ : Vector of dual variables associated with Constraint (4.1b). Let $\mu = \bar{\mu}$ be specific *values* associated with these dual variables.
- π : Vector of dual variables associated with Constraint (4.1c). Likewise, let $\pi = \bar{\pi}$ be specific values for these dual variables.
- π_0: Dual variable associated with Constraint (4.1d), with $\bar{\pi}_0$ being a specific value assumed by this variable.

The pricing subproblem, denoted by $\mathbf{SP}(\bar{\mu}, \bar{\pi}, \bar{\pi}_0)$, is formulated as the following 0-1 MIP:

$$\mathbf{SP}(\bar{\mu}, \bar{\pi}, \bar{\pi}_0)\text{: Min} \sum_{(i,j)\in E} c_{ij}e_{ij} + \sum_{i\in S}\sum_{k\in K} f_{ik}s_{ik} - \sum_{i\in S}\bar{\pi}_i x_i - \sum_{k\in K}\bar{\mu}_k y_k - \bar{\pi}_0 \tag{4.2a}$$

$$\text{subject to} \sum_{j\in S} e_{0j} = 1 \tag{4.2b}$$

$$\sum_{j\in N-\{i\}} e_{ij} = x_i, \quad \forall i \in S \tag{4.2c}$$

$$\sum_{i\in N-\{j\}} e_{ji} - \sum_{i\in N-\{j\}} e_{ij} = 0, \quad \forall j \in N \tag{4.2d}$$

$$q_j \geq q_i + \sum_{k\in K} d_k s_{jk} - 2Q(1 - e_{ij}) + Qe_{ij}, \quad \forall i, j \in S | i \neq j \tag{4.2e}$$

$$s_{ik} \leq x_i, \quad \forall i \in S, k \in K \tag{4.2f}$$

$$x_i \leq \sum_{k\in K} s_{ik}, \quad \forall i \in S \tag{4.2g}$$

$$\sum_{i\in S} s_{ik} = y_k, \quad \forall k \in K \tag{4.2h}$$

$$\sum_{k\in K} d_k s_{ik} \leq q_i \leq Qx_i, \quad \forall i \in S \tag{4.2i}$$

$$x, y, e, s \in \{0, 1\}, q \geq 0. \tag{4.2j}$$

The objective function (4.2a) minimizes the reduced cost of the constructed column. Constraints (4.2b) and (4.2c) respectively ensure that the central depot and any selected delivery site must have exactly one successor. Flow balance constraints are introduced in (4.2d). Constraint (4.2e) enforces subtour elimination constraints using the cumulative delivery made upon a visiting a particular delivery site. Constraint (4.2f) precludes the assignment of customers to a delivery site, unless the latter is selected, whereas Constraint (4.2g) requires at least one customer to be assigned to a selected delivery site. Constraint (4.2h) guarantees that any selected customer is assigned to exactly one delivery site. Constraint (4.2i) relates the cumulative delivery made upon visiting delivery site i to the total demand assigned to this site (as a lower bound) and to the total vehicle tour capacity (as an upper bound).

The pricing subproblem is NP-hard and computationally challenging for mid-sized problem instances solved using CPLEX. In fact, even if optimal customer assignments to delivery sites were *a priori* pre-specified, the subproblem would reduce to a prize-collecting TSP which is also known to be NP-hard. It is, therefore, crucial to seek an alternative solution methodology to the pricing subproblem in order to enable the development of effective branch-and-price algorithms for the VRDAP, as discussed in Section 4.

4.2.4 Branching Strategy

A best-first strategy is adopted in exploring the search tree and branching is conducted over routing and assignment arcs. Specifically, for each B&P tree node having a fractional solution, we first compute the flow of all routing arcs (arcs $(i, j) \in E$). The arc (i, j) having a most fractional flow (nearer to 0.5) is selected for branching and two new child nodes are generated accordingly by introducing arc (i, j) in one child and forbidding it in the other. If no routing arcs with fractional flow are found, the branching scheme uses customer assignment arcs (arcs $(k, j) \in E^{'}$). Similarly, a most fractional arc (k, j) is detected and two child nodes are created; in one child node, we assign customer k to site j, whereas in the other, the assignment of k to j is forbidden.

4.3. Dynamic Programming Algorithm for the Pricing Subproblem

We propose in this section a specialized DP algorithm for the CG pricing subproblem. In Section 4.3.1, we first delineate the overall DP algorithm. Thereafter, we present in Section 4.3.2 a label extension scheme and a label dominance proce-

Table 4.1: Original customer assignment cost matrix (f)

$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline a & 5 & 10 & 20 & 30 & 60 & 50 \\ b & 12 & 5 & 10 & 20 & 50 & 40 \\ c & 25 & 20 & 5 & 20 & 55 & 45 \\ d & 20 & 5 & 10 & 15 & 35 & 30 \\ e & 35 & 30 & 10 & 15 & 45 & 45 \\ f & 40 & 30 & 20 & 5 & 40 & 45 \\ g & 40 & 25 & 20 & 5 & 25 & 30 \\ h & 35 & 20 & 30 & 20 & 10 & 10 \\ i & 30 & 15 & 25 & 25 & 15 & 8 \\ j & 45 & 35 & 45 & 40 & 10 & 5 \\ k & 40 & 30 & 50 & 45 & 15 & 5 \end{array}$$

Table 4.2: Original routing cost matrix (c)

$$\begin{array}{c|ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 0 & - & 10 & 10 & 25 & 35 & 25 & 15 \\ 1 & 10 & - & 12 & 20 & 35 & 35 & 30 \\ 2 & 10 & 12 & - & 15 & 20 & 30 & 20 \\ 3 & 25 & 20 & 15 & - & 12 & 40 & 35 \\ 4 & 35 & 35 & 20 & 12 & - & 30 & 35 \\ 5 & 25 & 35 & 30 & 40 & 30 & - & 12 \\ 6 & 15 & 30 & 20 & 35 & 35 & 12 & - \end{array}$$

dure. Various heuristic implementations of the proposed DP algorithm are discussed in Section 4.3.3. Throughout this section, we consider a VRDAP instance with a set of 6 candidate delivery sites $S = \{1, 2, \ldots, 6\}$ and 11 customers $K = \{a, b, \ldots, k\}$. The vehicle capacity is $Q = 8$ and an equal demand of 1 is considered for all customers. The assignment and routing cost matrices are given in Tables 4.1 and 4.2, respectively.

4.3.1 Overall Exact DP Approach

The elementary shortest path problem with resource constraints (ESPPRC) arises in the pricing subproblems of the VRP with time-windows and the capacitated VRP, among others. Although the ESPPRC itself is NP-hard in the presence of negative cost cycles (Dror 1994), DP algorithms as in Feillet et al. (2004) and similar works provide an exact solution method to such problems. A key difficulty is to ensure the elementariness of the constructed shortest path, i.e., the requirement that any node be visited at most once. If this requirement is relaxed, the problem reduces to a shortest path problem with resource constraints (SPPRC) and can be solved using pseudo-polynomial algorithms. However, this computational convenience comes at the expense of producing weaker lower bounds. Several papers develop cycle-elimination approaches to overcome this drawback. For example, Christofides et al. (1981) propose a method to eliminate cycles of length two, whereas Irnich and Villeneuve (2006) investigate the elimination of k-cycles (with $k \geq 2$). The computational burden associated with the elimination of cycles of size four or more is, however, not accompanied by a commensurate improvement in the lower bounds (Fukasawa et al. 2006). Recently, Martinelli et al. (2014) enhanced the use of *ng-routes*, originally proposed by Baldacci et al. (2011), by exploring a decremental state-space relaxation technique (see also Boland et al. 2006 and Righini and Salani 2008 for related works). In our experience, however, a specialized labeling type DP algorithm using an adaptation of the work by Feillet et al. (2004), as detailed next, has been found to be more advantageous for the VRDAP than adapting these more recent approaches.

The ESPPRC for VRDAP is defined over the network $G = (N \cup K,\ E \cup E^{'})$ where $N \cup K$ form a set of nodes comprising the depot, delivery sites and customers, and where $E = \{(i,j) |\, i \neq j \in N\}$, and $E^{'} = \{(k,j) |\, k \in K, j \in S\}$ constitute arcs. Recall that each customer $k \in K$ has a demand d_k and the original costs of arcs $(i,j) \in E$ and $(k,j) \in E^{'}$ are c_{ij} and f_{kj}, respectively. To incorporate the dual values associated with Constraints (2.1b)-(2.1d), respectively denoted by $\bar{\pi}$, $\bar{\mu}$, and $\bar{\pi}_0$, we define the following new arc costs:

$$w_{ij} = c_{ij} - (\bar{\pi}_i + \bar{\pi}_j)/2, \quad \forall (i,j) \in E \tag{4.3a}$$

$$w_{kj} = f_{kj} - \bar{\mu}_k, \quad \forall (k,j) \in E^{'}. \tag{4.3b}$$

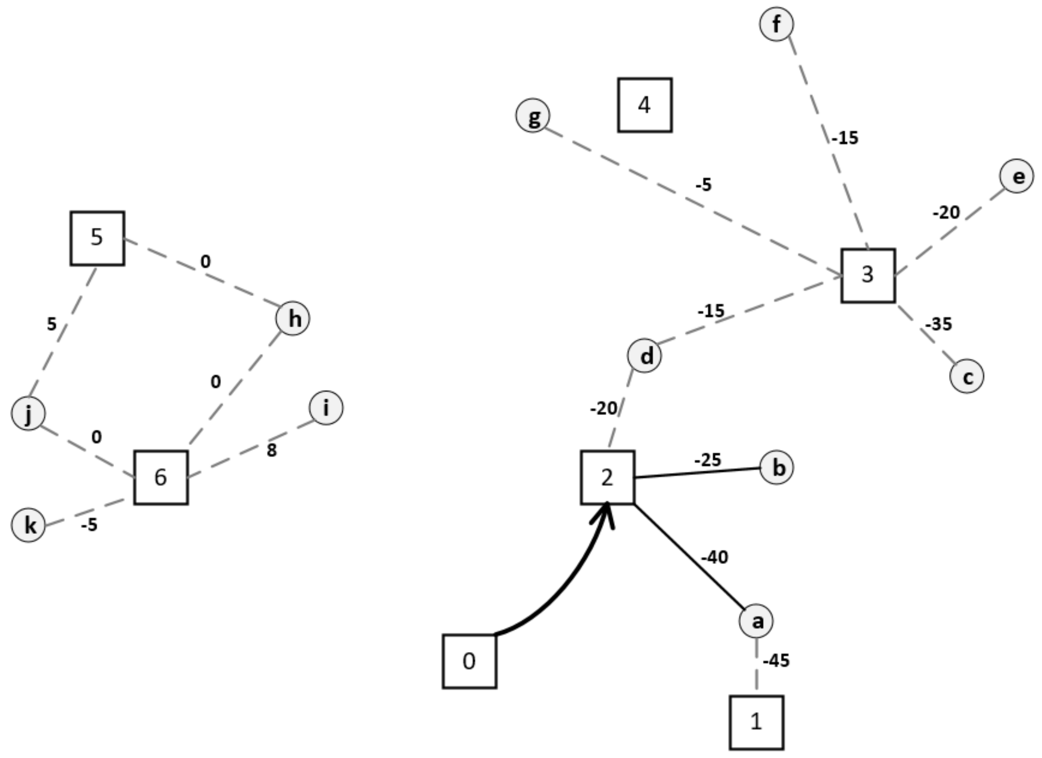

Figure 4.2: Solid lines represent routing and assignment decisions made thus far in subproblem

Because any delivery site i along the constructed route has one incoming and one outgoing arc, the contribution of $\bar{\pi}_i$ is split in half between these two arcs in (4.3a). In this manner, the contribution of $\bar{\pi}_0$ (associated with constraint (2.1d)) is also incorporated in the cost of arcs entering and leaving the depot (node 0). In our illustrative example, Tables 4.5 and 4.6 respectively show the updated assignment and routing cost matrices after incorporating dual values given in Tables 4.3 and 4.4 into original cost matrices. Figure 4.2 shows the associated network of this instance along with some customer assignment arcs and their associated costs. The objective of the subproblem is to find the minimum cost feasible tour in this network (using the updated cost matrices in Tables 4.5 and 4.6).

The DP algorithm for ESPPRC employs the notion of labels in order to identify distinct paths that are extended to particular nodes in the network. Starting at

Table 4.3: Dual values associated with constraints (4.1b)

$\bar{\mu}_a$	$\bar{\mu}_b$	$\bar{\mu}_c$	$\bar{\mu}_d$	$\bar{\mu}_e$	$\bar{\mu}_f$	$\bar{\mu}_g$	$\bar{\mu}_h$	$\bar{\mu}_i$	$\bar{\mu}_j$	$\bar{\mu}_k$
50	30	40	25	30	35	25	10	0	5	10

Table 4.4: Dual values associated with constraints (4.1c); $\bar{\pi}_0$ is the dual value of (4.1d)

$\bar{\pi}_0$	$\bar{\pi}_1$	$\bar{\pi}_2$	$\bar{\pi}_3$	$\bar{\pi}_4$	$\bar{\pi}_5$	$\bar{\pi}_6$
15	5	10	0	15	0	0

Table 4.5: Assignment cost matrix after applying dual values

$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline a & -45 & -40 & -30 & -20 & 10 & 0 \\ b & -18 & -25 & -20 & -10 & 20 & 10 \\ c & -15 & -20 & -35 & -20 & 15 & 5 \\ d & -5 & -20 & -15 & -10 & 10 & 5 \\ e & 5 & 0 & -20 & -15 & 15 & 15 \\ f & 5 & -5 & -15 & -30 & 5 & 10 \\ g & 15 & 0 & -5 & -20 & 0 & 5 \\ h & 25 & 10 & 20 & 10 & 0 & 0 \\ i & 30 & 15 & 25 & 25 & 15 & 8 \\ j & 40 & 30 & 40 & 35 & 5 & 0 \\ k & 30 & 20 & 40 & 35 & 5 & -5 \end{array}$$

Table 4.6: Routing cost matrix after applying dual values

$$\begin{array}{c|ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 0 & - & 0 & -2.5 & 17.5 & 20 & 17.5 & 7.5 \\ 1 & 0 & - & 4.5 & 17.5 & 25 & 32.5 & 27.5 \\ 2 & -2.5 & 4.5 & - & 10 & 7.5 & 25 & 15 \\ 3 & 17.5 & 17.5 & 10 & - & 4.5 & 40 & 35 \\ 4 & 20 & 25 & 7.5 & 4.5 & - & 22.5 & 27.5 \\ 5 & 17.5 & 32.5 & 25 & 40 & 22.5 & - & 12 \\ 6 & 7.5 & 27.5 & 15 & 35 & 27.5 & 12 & - \end{array}$$

node 0 (the depot), labels are iteratively extended to non-visited nodes, provided that this path extension is feasible (with regard to the tour capacity, Q) and yields a non-dominated label (where cost considerations relate to the reduced cost of the constructed column). As paths get extended, new delivery sites and customers are added. It is important to dynamically monitor the residual capacity of a label to ensure feasibility and its updated cost to detect sub-optimality using dominance rules. This curtails the proliferation of unattractive labels and accelerates the convergence to an optimal solution.

First, the algorithm employs a preprocessing routine that eliminates delivery sites and customers that would have a non-negative contribution to the reduced cost, thereby yielding suboptimal solutions to the pricing subproblem. To this end, in lieu of the original sets of sites (S) and customers (K), we operate over the following reduced subsets S^R and K^R:

- $S^R = \{s \in S : -\bar{\pi}_s + \sum_{k \in K} \min\{0, w_{ks}\} < 0\}$;
- $K^R = \{k \in K : \min_{s \in S} w_{ks} < 0\}$.

In our illustrative example, the best contribution of delivery site 5 is 0 ($\bar{\pi}_5 = 0$, $w_{k5} \geq 0 \; \forall k \in K$) and hence it is not included in S^R. For customer i, $w_{is} \geq 0 \; \forall s \in S$ and, therefore, assigning it to any delivery site would not improve the solution cost. Similarly, customers h and j are removed and $K^R = \{a, b, c, d, e, f, g, k\}$.

Each label, denoted by $\boldsymbol{\mathcal{L}} = (P_\ell, A_\ell, C_\ell, Q_\ell)$, comprises the following elements: (i) $P_\ell = (0, v_1, \ldots, v_\ell)$ an ordered set of delivery sites visited; (ii) $A_\ell = (\varnothing, a_1, \ldots, a_\ell)$ an ordered set of clustered customers respectively associated with each delivery site included in the label (e.g., a_i is the set of customers assigned to delivery site v_i); (iii) $C_\ell = w_{0,v_1} + \sum\limits_{i=1}^{\ell-1} w_{v_i,v_{i+1}} + \sum\limits_{i=1}^{\ell} \sum\limits_{k \in a_i} w_{k,v_i}$, cost of the path; and (iv) $Q_\ell = \sum\limits_{i=1}^{\ell} \sum\limits_{k \in a_i} d_k$, the total customer demand assigned to this label. Note that, in order for label $\boldsymbol{\mathcal{L}}$ to represent a feasible elementary shortest path the following conditions must hold:

Algorithm 6 ESPPRC-VRDAP

```
1: $H_v = \varnothing \quad \forall v \in S$
2: $H_0 = \bar{H}_0 = \{((0), (\varnothing), 0, 0)\}$
3: $\Gamma = \{0\}$
4: while $\Gamma \neq \varnothing$ do
5:     Select a site $u$ from $\Gamma$ and $\Gamma \leftarrow \Gamma \setminus \{u\}$
6:     for all $v \in S^R$ do
7:         $F_v = \varnothing$
8:         for all $L \in \bar{H}_u \mid v \notin (\sigma_\ell \cup \tilde{\sigma_\ell})$ do
9:             $F_v \leftarrow F_v \cup$ GET-LABELS$(L, v)$
10:        end for
11:        $F_v \leftarrow$ ADD$(H_v, F_v)$
12:        if $F_v \neq \varnothing$ then
13:            $H_v \leftarrow H_v \cup F_v$
14:            $\bar{H}_v \leftarrow F_v \cup \bar{H}_v$
15:            $\Gamma \leftarrow \Gamma \cup \{v\}$
16:        end if
17:    end for
18:    $\bar{H}_u \leftarrow \varnothing$
19: end while
```

$Q_\ell \leq Q$ (capacity is not violated), $\cap_{i=1}^{\ell} a_i = \varnothing$ (assigned customers form disjoint subsets), and any delivery site along the path P_ℓ is visited exactly once. Further, let σ_ℓ and χ_ℓ be the sets of sites and customers that are already visited by $\mathcal{L}$, respectively. Let $\tilde{\sigma_\ell}$ and $\tilde{\chi_\ell}$ respectively be the sets of sites and customers that, although not included in $\mathcal{L}$, should not be considered for inclusion in any future extension of this label because they would yield dominated labels (These sets are computed in an algorithm that we refer to as GET-LABELS in Section 4.3.2.)

In our illustrative example, let $\mathcal{L}_1$ be the label corresponding to a partial solution constructed by extending the depot (0) to delivery site 2 and assigning to it the subset of customers $\{a, b\}$. In Figure 4.2, the solid lines represent the arcs included in this label. For label $\mathcal{L}_1$, $P_{\ell_1} = (0, 2)$, $A_{\ell_1} = (\varnothing, \{a, b\})$, $C_{\ell_1} = -67.5$, $Q_{\ell_1} = 2$, $\sigma_{\ell_1} = \{2\}$, $\chi_{\ell_1} = \{a, b\}$.

Algorithm 1 ESPPRC-VRDAP finds an optimal elementary shortest path starting and ending at the depot (node 0) and passing through selected delivery sites. The

optimal solution corresponds to a label $\mathcal{L}^*$ with a minimum cost $C_{\ell^*} + w_{v_{\ell^*},0}$. To discuss the DP algorithm, the following notation is introduced:

- H_v: Current set of non-dominated labels $\mathcal{L}$ such that the last delivery site along the path is v.
- F_v: Set of all labels newly extended to v.
- $\bar{H}_v$: Set of labels in H_v that are not extended yet. This is an auxiliary set that is used in Algorithm 1 as discussed next.
- Γ: Set of all sites v for which there is at least one label that is not extended yet ($\bar{H}_v \neq \varnothing$).

Algorithm 1 ESPPRC-VRDAP first creates an initial label at the depot (i.e., node 0) which is added to the set Γ of active sites that have at least one label that is yet to be extended. Having selected a site u in Γ (using the while statement), the algorithm loops over all delivery sites $v \in S^R$ and considers extending any label $\mathcal{L}$ in the set $\bar{H}_u$ to include node v, provided that this extension is feasible, i.e., $v \notin \sigma_\ell$, and is not readily known to yield a dominated label, i.e., $v \notin \tilde{\sigma_\ell}$. The details of the path extension scheme (GET-LABELS) are discussed in Section 4.3.2. This results in a set of labels newly extended to site v, denoted by F_v. The function ADD (line 11) applies dominance rules for the stored set of non-dominated labels for v, H_v, and the newly extended labels to v, F_v. As a result, presently dominated labels in H_v and F_v are eliminated, set $\bar{H}_v$ is updated, and delivery site v is added to Γ as an active site which should be considered for future label extension. This iterative procedure continues until the set Γ is exhausted and an optimal solution is obtained. Appendix C illustrates some steps of Algorithm 3 over a numerical example.

4.3.2 Path Extension Scheme

A key element of the DP algorithm is the path extension scheme which is referred to as GET-LABELS($\mathcal{L}, v_\ell$) in Algorithm 1 ESPPRC-VRDAP. This creates all the feasible labels that can be extended from label $\mathcal{L}$ by adding a new delivery site v_ℓ to the route and assigning to it a subset of customers. Recall that K^R is the reduced set of customers after preprocessing. Let $K^{'} = K^R \setminus \chi_\ell$, where χ_ℓ is the set of customers already assigned to ℓ. The function GET-LABELS finds all the subsets of $K^{'}$ having an aggregate demand that does not exceed the residual capacity of vehicle, (i.e. $Q - Q_l$). A new label is created by assigning the customers in each subset to v_ℓ.

The number of feasible subsets generated from $K^{'}$ can be prohibitively large and it is important to reduce the size of $K^{'}$ by removing customers that would yield a dominated label, if they were assigned to site v_ℓ. We propose three rules for detecting such customers that greatly contribute to the efficacy of the algorithm.

- **Rule 1: Eliminating Unattractive Customers for v_ℓ**

  Consider $K^{'} = K^R \setminus \chi_\ell$ and $S^{'} = S^R \setminus (\sigma_\ell \cup \tilde{\sigma_\ell})$. All customers $k \in K^{'}$ such that $w_{k,v_\ell} \geq 0$ are removed from $K^{'}$, because their assignment to v_ℓ would worsen the solution. Further, for any customer $k \in K^{'}$, if there exists a site $v_i \in \sigma_\ell$ such that $w_{k,v_\ell} \geq w_{k,v_i}$, then customer k is removed from $K^{'}$. In fact, if we assigned k to v_ℓ, the resulting label would be dominated by a similar label in which k is removed from v_ℓ and assigned to v_i.

  To illustrate, consider extending $\mathcal{L}_1$ in our example to delivery site 3. Customer k is removed from $K^{'}$ since $w_{k,3} = 40$. Also, note that $w_{d,3} > w_{d,2}$ therefore, customer d is also removed from $K^{'}$, reducing the set of remaining customers $K^{'}$ to $\{c, e, f, g\}$.

- **Rule 2: Using Unassigned Customers for which v_ℓ is Closest**

  Let Ψ be the subset of customers in $K^{'}$ such that their closest delivery site in $S^{'}$, in the sense of the f-values, is v_ℓ. Observe that any customer in Ψ, if it is not assigned to v_ℓ itself, it will never be assigned to any future delivery site that is added to $\mathcal{L}$ after v_ℓ. (Otherwise, the resulting label would be dominated by a similar label with the only difference that this customer is assigned to v_ℓ.) Also, let Ω be the set of all customers $o \in \Psi$ for which there exists a customer k included in label $\mathcal{L}$ such that k is assigned to v_i, $d_k \geq d_o$, and $w_{k,v_i} \geq w_{o,v_\ell}$. If $o \in \Omega$ is not assigned to v_ℓ, then o will never be assigned to any label $\mathcal{L}^{'}$ that stems from $\mathcal{L}$. Furthermore, by removing k from site v_i and assigning o to v_ℓ instead, we can construct a better label. As a result, for the new label $\mathcal{L}^{'}$ to be non-dominated, assign all customers in Ω to v_ℓ and remove all the members of Ω from $K^{'}$ and Ψ. Note, of course, that if the inclusion of Ω was not feasible due to the tour capacity, this label would be dominated and should not be given further consideration.

  To illustrate using our example, site 3 in the partial path is the closest site of customers c and e. Therefore, $\Psi = \{c, e\}$. Therefore, these customers would be either assigned to site 3 or not at all in any future extension of $\mathcal{L}_1$. Also, noting that $d_c \leq d_b$ and $w_{c,3} \leq w_{b,2}$, customer c has a better contribution to the solution and has a smaller demand than customer b. Therefore, $\Omega = \{c\}$ and customer c is assigned to site 3 and $K^{'} = \{e, f, g\}$, $\Psi = \{f\}$.

- **Rule 3: Identifying So-Called Inferior Customers**

  A customer $k \in K^{'}$ is deemed to be *inferior* to $o \in \Psi$, if $d_k \geq d_o$ and $w_{k,v_\ell} \geq w_{o,v_\ell}$. If customer o is not assigned to v_ℓ, then none of its inferior customers can be assigned to v_ℓ (otherwise, this would result in a dominated label). In the next step of GET-LABELS, inferior customers are determined for every $o \in \Psi$. This information helps reduce the number of feasible customer subsets to be created in order to extend the label, as discussed next. For example, among remaining customers of $K^{'}$, f and g are inferior to e. Therefore, if e is not assigned to site 3, neither of f and g can be assigned to it.

4.3.2.1 Creating Feasible Customer Subsets

Because all customers in Ω were assigned to v_ℓ at an earlier point, note that the residual capacity is $Q - (Q_\ell + \sum_{k \in \Omega} d_k)$. We then determine all subsets of $K^{'}$ such that their aggregate demand does not exceed $Q-(Q_\ell+\sum_{k\in\Omega} d_k)$. Furthermore, when computing feasible subsets of $K^{'}$, if $o \in \Psi$ is not in a subset, then none of its inferior customers can be included in this subset as well. For our example, $K^{'} = \{e, f, g\}$, $\Psi = \{e\}$, $\Omega = \{c\}$. For each of the customers subsets $\{c\}$, $\{c, e\}$, $\{c, e, f\}$, $\{c, e, g\}$, $\{c, e, f, g\}$ a new label is created. Note that c must be assigned to 3 and customers f and g are inferior to e. Before using Rules 1-3, $K^{'}$ had 6 members, resulting in 64 possible customer subsets; that is, the application of these rules precluded the generation of 59 dominated labels.

For each resulting subset of customers a_ℓ, a new label $\mathcal{L}^{'}$ is created by extending $\mathcal{L}$ to v_ℓ and assigning to v_ℓ customers of a_ℓ along with customers in Ω. The attributes of label $\mathcal{L}^{'}$ are specified as follows:

- With $P_\ell = (0, v_1, \ldots, u)$ being the sequence of sites visited by $\mathcal{L}$, the partial path associated with $\mathcal{L}^{'}$ is $P_{\ell'} = (0, v_1, \ldots, u, v_\ell)$.

- With $A_\ell = (\varnothing, a_1, \ldots, a_u)$ being the ordered set of clustered customers assigned to sites in P_ℓ, we set $A_{\ell'} = (\varnothing, a_1, \ldots, a_u, a_\ell \cup \Omega)$ for label $\mathcal{L}^{'}$.

- The set of delivery sites and customers visited by $\mathcal{L}^{'}$ are $\sigma_{\ell'} = \sigma_\ell \cup \{v_\ell\}$ and $\chi_{\ell'} = \chi_\ell \cup (a_\ell \cup \Omega)$, respectively.
- The cost of the new label is $C_{\ell'} = C_\ell + w_{u,v_l} + \sum\limits_{k \in a_\ell \cup \Omega} w_{k,v_\ell}$ and its consumed capacity is $Q_{\ell'} = Q_\ell + \sum\limits_{k \in a_\ell \cup \Omega} d_k$.

The final step of function GET-LABELS is to update $\tilde{\sigma}_{\ell'}$ which is the set of sites in S^R that although not included in $\mathcal{L}^{'}$, should not be considered for the extension of this label because they would yield dominated labels. In fact, for customer k already assigned to a site s, any site $\bar{s}$ that is closer to k than s (i.e. $w_{k,\bar{s}} < w_{k,s}$), is a member of $\tilde{\sigma_{\ell'}}$. The reason is that any future extension of $\mathcal{L}^{'}$ that visits $\bar{s}$, is dominated by a similar label in which instead of s, k is assigned to $\bar{s}$. In our example, let $\mathcal{L}_2$ be a label obtained from extending $\mathcal{L}_1$ to site 3 and assigning to site 3 customers $\{c, e, g\}$. For this label, $\tilde{\sigma}_{\ell_2} = \{1, 4\}$. In fact, visiting site 1 in any future extension of $\mathcal{L}_2$ would make the assignment of customer a to site 2 a suboptimal decision and, hence, the resulting label would be dominated. The same logic applies to site 4; its inclusion in any future extension would make the assignment of customer g to site 3 suboptimal.

At last, we construct $\tilde{\chi}_{\ell'}$, the set of customers that although not assigned to any site yet, should not be considered for assignment in any future extension of $\mathcal{L}^{'}$ because they would yield dominated labels. Recall that Ψ is the set of customers such that their assignment to any site after v_ℓ would result in a dominated label. Let $\varsigma = \Psi \setminus (a_\ell \cup \Omega)$ and add ς to $\tilde{\chi}_{\ell'}$ ($\tilde{\chi}_{\ell'} \leftarrow \tilde{\chi}_{\ell'} \cup \varsigma$). For example, in label $\mathcal{L}_2$, $\tilde{\chi}_{\ell_1} = \{d\}$ since d is not assigned to its closest site (i.e., site 2), it cannot be assigned to any other site in future extensions of $\mathcal{L}_2$. Note that, although Rule 1 detects such customers and avoids their inclusion in future extensions, the set $\tilde{\chi}_\ell$ is used to strengthen the dominance rule presented next.

4.3.2.2 Dominance Rule

In Algorithm 1 ESPPRC-VRDAP, the function ADD(H_v, F_v) removes all dominated labels in H_v (previously extended labels to site v) and F_v (recently extended labels to site v) and returns all non-dominated labels in F_v. As such, $\forall \mathcal{L}^* \neq \mathcal{L} \in F_v$, $\mathcal{L}^*$ is said to dominate $\mathcal{L}$ if the following conditions hold: (i) $C_{\ell^*} \leq C_\ell$; (ii) $Q_{\ell^*} \leq Q_\ell$; (iii) $\sigma_{\ell^*} \subseteq \sigma_\ell \cup \tilde{\sigma}_\ell$; and (iv) $\chi_{\ell^*} \subseteq \chi_\ell \cup \tilde{\chi}_\ell$. Dominated labels are removed from F_v and H_v by applying Algorithm 7.

Algorithm 7 DOMINANCE$(\mathcal{L}^*, \mathcal{L})$ – Checks if $\mathcal{L}^*$ dominates $\mathcal{L}$

```
1: **if** $C_{\ell^*} > C_\ell$ or $Q_{\ell^*} > Q_\ell$ **then**
2:     return FALSE
3: **end if**
4: **if** $(\sigma_{\ell^*} \subseteq \sigma_\ell \cup \tilde{\sigma}_\ell)$ AND $(\chi_{\ell^*} \subseteq \chi_\ell \cup \tilde{\chi}_\ell)$ **then**
5:     return TRUE
6: **else**
7:     return FALSE
8: **end if**
```

4.3.3 Heuristic DP Procedures

In this section, we propose six heuristic versions of the DP algorithm (Algorithm 1 ESPPRC-VRDAP) which, in our experience, play an important role in identifying near-optimal solutions to the pricing subproblem in short CPU times. This, in turn, significantly accelerates the overall B&P algorithm. Our strategy has been to invoke these heuristic variants, until no column with a negative reduced cost is found. In the latter case, we resort to using the exact DP approach to either reveal such a column, when it exists, or to establish LP optimality at a node of the B&P tree. The names of the heuristic DP variants presented next follow this logic: (i) (R) refers to a relaxed dominance rule, as opposed to (NR), not relaxed; and (ii) (EZ) refers to a simplified (easy) generation of customer subsets to be appended to a delivery site, whereas (H) refers to an exhaustive (hard) enumeration of such subsets.

H1-R: In Heuristic H1-R, any customer $k \in K^R$ is assigned to its nearest delivery site and the pricing subproblem reduces to that of a capacitated VRP (CVRP). We use a relaxed (heuristic) dominance rule whereby a label $\mathcal{L}^*$ dominates $\mathcal{L}^{'}$, if $C_{\ell^*} \leq C_{\ell'}$ and $Q_{\ell^*} \leq Q_{\ell'}$ (i.e., we do not check the second condition in Algorithm 7 DOMINANCE).

H1-NR: Heuristic H1-NR is similar to Heuristic H1-R except that it employs the exact dominance rule in Algorithm 7.

H2-R: In Heuristic H2-R, we use the relaxed dominance rule as in Heuristic H1-R, but with a different customer assignment scheme. Let v be a delivery site that is considered for inclusion in a label and to which we would like to assign customers, subject to a residual capacity $\bar{q}_\ell = Q - Q_\ell$. Instead of calling GET-LABELS, Heuristic H2-R proceeds as follows. First, let $K^{'} = K^R \setminus \chi_\ell$. Then, we remove any customer $k \in K^{'}$ such that v is not amongst its first ϵ nearest sites. We also use Rules 1-3 of function GET-LABELS to further reduce $K^{'}$. For each customer $k \in K^{'}$, we compute $w_{k,v}/d_k$ which can be seen as the reduction in cost, for any one unit of increase in vehicle load, after assigning k to v. Heuristic H2-R uses this quantity as a measure of attractiveness of assigning k to v. Let $\{k_{[1]}, k_{[2]}, \dots, k_{[|K'|]}\}$ be the set of all customers in $K^{'}$ sorted in the ascending order of their $w_{k,v}/d_k$ values. The first label is constructed by assigning the most attractive customer $k_{[1]}$ to v . Then, another label is constructed by assigning first two most attractive customers $\{k_{[1]}, k_{[2]}\}$. This is continued until i^{th} label is constructed by assigning $\{k_{[1]}, ..., k_{[i]}\}$ to v and adding $k_{[i+1]}$ to this set will violate vehicle capacity ($i = \text{argmin}\{h \,|\, \sum_{j=1}^{h} d_{v_{[j]}} > \bar{q}_\ell\}$).

H2-NR/EZ: Heuristic H2-NR/EZ employs the customer assignment scheme in Heuristic H2-R, but with the exact dominance rule.

H2-NR/H: Heuristic H2-NR/H is similar to H2-R: First, the set $\{k_{[1]}, k_{[2]}, \ldots k_{[i-1]}\}$ is determined, but all its customer subsets are then generated and considered for potential assignment to site v (as opposed to considering $i-1$ sets only, i.e. $\{k_{[1]}\}$, $\{k_{[1]}, k_{[2]}\}$, …, $\{k_{[1]}, k_{[2]}, \ldots k_{[i-1]}\}$). The exact dominance rule is also employed here.

HR: In lieu of using sets S^R and K^R defined in Section 4.1, Heuristic HR considers, for any site s, the subset of customers for which site s is among their ϑ nearest delivery sites.

In our implementation, we trigger the aforementioned heuristics in the following order. First, we activate Heuristic HR. Then, we call Heuristic H1-R to solve the subproblem; if it fails to find an attractive column, Heuristic H1-NR is invoked. Again, if the latter fails, Heuristics H2-R, H2-NR/EZ and H2-NR/H are sequentially triggered in turn, as necessary. If any of these heuristics is successful, in addition to the best column it finds, we consider adding up to 20 columns with the most negative reduced costs (as available) to the RMP. However, when all heuristics fail to find a column with a negative reduced cost, Heuristic HR is deactivated and the heuristics are triggered with an exact reduction scheme. If they fail again, the exact DP (Algorithm 6) is invoked in which case either an attractive column is found or LP optimality is proven (at the current B&P tree node).

4.4. Computational results

In this section, we evaluate the computational performance of the B&P algorithm, which we implemented using C# under Visual Studio, and compare it against solving the original MIP model using the branch-and-cut algorithm of CPLEX 12.5. All runs were performed with a time limit of 3600 CPU seconds on a Windows 7 professional 64-bit operating system with an Intel Core i7-2600 CPU with 3.40 GHz and

12 GB RAM desktop. We also compare the overall performance of the exact solution of the B&P algorithm and its root-node incumbent integer solution against solutions produced using the CG approach in Ghoniem et al. (2013) and the Benders decomposition approach in Solak et al. (2014). This numerical comparison is possible noting that the aforementioned approaches were tested by the authors using comparable hardware/software settings.

4.4.1 Description of Problem Instances

Our test-bed comprises 60 challenging problem instances that were randomly generated using the data generation scheme in Ghoniem et al. (2013) where instances are based on the operations of a food bank in the Southeastern US. Instances were generated with the following characteristics:

- Number of delivery sites and customers: $|S| = 10$, 20, or 25 and $|K| = 20$, 30, 40, or 50. For each $(|S|, |K|)$ combination, five instances are reported, resulting in a total of 60 instances.

- Customer and delivery site locations: The coordinates of customers and delivery sites were randomly generated using a uniform distribution in a two-dimensional Euclidean space where the distance between the central depot, which serves as the origin, and customers/delivery sites varies between 25 to 75 miles. (It is assumed that customers that are within 25 miles from the depot follow a different delivery pattern and may directly collect their demand from the depot.)

- Customer demand: The demand of each customer, measured in pallets, was randomly generated between 1 and 5 pallets with the following probabilities: "$P(d_k = 1)$" $= 0.5$, "$P(d_k = 2)$" $= 0.2$, "$P(d_k = 3)$" $= 0.1$, "$P(d_k = 4)$" $= 0.1$, and "$P(d_k = 5)$" $= 0.1$.

- The vehicle capacity is set to $Q = 25$ (pallets).

- The number of vehicle tours considered, $|V|$: Noting that $m = \left\lceil (\sum_{k=1}^{|K|} d_k)/Q \right\rceil$ is a lower bound on the number of vehicle tours needed for serving the total demand, we set $|V| = m + 2$ in our test-bed.

4.4.2 Performance of B&P Algorithm vs. CPLEX

Table 4.7 reports our computational results for solving the base MIP formulation (in Appendix A) using CPLEX 12.5 vs. solving the set partitioning reformulation using our proposed B&P algorithm. Columns 1 and 2 respectively specify the size of the instance, $(|S|, |K|)$, and its reference. For CPLEX, Columns 3-5 respectively report the objective value obtained by the solver, be it optimal or just an upper bound, the CPU time in seconds, and the solver optimality gap at termination if an instance is not solved within a time limit of 3600 CPU seconds. The remainder of Table 4.7 relates to the B&P algorithm. Columns 6-9 focus on the root-node performance of the B&P algorithm and report the following: (i) The lower bound (LB) achieved by column generation; (ii) upper bound (UB) corresponding to the best integer solution identified at the root-node by solving the RMP as a 0-1 problem using CPLEX; (iii) the optimality gap (%) at the root-node comparing the aforementioned UB and LB; and (iv) the CPU time (s) consumed at the root-node. Columns 10-14 summarize the overall performance of the B&P algorithm with respect to the following: (i) The optimal objective value or the upper bound obtained when an instance is not solved to provable optimality within 3600 CPU seconds; (ii) the optimality gap at termination or within a time limit of 3600 CPU seconds; (iii) total CPU time (s); (iv) number of B&P nodes explored; and (v) the relative computational savings (in CPU time) achieved by the B&P algorithm over the B&B/C algorithm in CPLEX.

The results in Table 4.7 reflect the computational challenges posed by this VRDAP and the substantial CPU time savings achieved by the proposed DP-based branch-and-price algorithm over a commercial solver such as CPLEX. In fact, CPLEX failed to solve the MIP model in Appendix A for 48/60 instances to provable optimality within one CPU hour and terminated with optimality gaps ranging from 4.4% to over 52% and averaging 25% over these instances. In contrast, the B&P algorithm produced an optimal solution for 56/60 instances in our test-bed and yielded an optimality gap of about 2% on average for the 4/60 instances it did not solve to provable optimality within one CPU hour. Of the 56/60 instances it solved to provable optimality, the B&P algorithm solved 23 instances at the root-node itself. Moreover, even for the smaller instances that CPLEX solved within one CPU hour, the B&P algorithm achieved remarkable savings in CPU time. Over our entire test-bed, the B&P algorithm achieved computational savings in CPU time over CPLEX ranging from 16% to nearly 100% and averaging 86%. Instance 5, with $(|S|, |K|) = (25, 20)$, is quite striking: CPLEX exhibited an optimality gap of 20.5% after one CPU hour, whereas the B&P algorithm produced an optimal solution at its root-node in 1.8 CPU second. With regard to the inadequacy of CPLEX for these instances, we note the following:

- For instances with a smaller number of customers, e.g. $(|S|, |K|) = (20, 20)$ or $(|S|, |K|) = (25, 20)$, CPLEX often reported a large optimality gap, although its incumbent solution is observed to be either optimal or near-optimal when compared against the B&P algorithm results. This is indicative of the weakness of the LP relaxation of the base MIP formulation for this problem. As such, CPLEX is expending a great deal of effort to close the gap between its weak LP bounds and its optimal/near-optimal MIP incumbent solution. This drawback is, of course, circumvented in the B&P algorithm, as column generation LP bounds are usually very strong.

- As the number of customers increases, the performance of CPLEX deteriorates. In addition to the aforementioned weakness of the LP relaxation, the solver experiences difficulties in identifying high quality MIP solutions within one CPU hour (see for example, the results of CPLEX for instances with $(|S|, |K|) = (20, 50)$ or $(|S|, |K|) = (25, 50)$).

4.4.2.1 Usefulness of DP Variants

In Table 4.8, we detail for each instance the total number of subproblems solved by DP and the associated total CPU time. We also provide a specific breakdown for these two metrics across the five heuristic DP variants (H1-R, H1-NR, H2-R, H2-NR/EZ, or H2-NR/H), as described in Section 4.3.3, and the exact DP algorithm. The results highlight the relative and collective benefit of the five DP heuristics for the B&P algorithm. In summary, averaging the results in Table 4.8 over the entire test-bed, we note the following:

- Heuristic H1-R (with a relaxed dominance rule) solved 51% of the subproblems in only 3.2% of the total CPU time devoted to solving the subproblems.
- Heuristic H1-NR (with an exact dominance rule) solved 20% of the subproblems in about 0.1% of the total CPU time.
- Heuristic H2-R (with a relaxed dominance rule) solved 20% of the subproblems in about 35.2% of the total CPU time.
- Heuristic H2-NR/EZ (with an exact dominance rule and an "easy" customer assignment) solved 5% of the subproblems in 30% of the total CPU time.
- Heuristic H2-NR/H (with an exact dominance rule and a more elaborate customer assignment scheme) solved 3% of the subproblems in 5.6% of the total time.
- The exact DP algorithm solved 1% of the subproblems in about 25.9% of the total CPU time.

That is, in only 1% of the subproblems did the B&P algorithm require the use of the exact DP algorithm. This usually served the purpose of identifying a new attractive column that was beyond the reach of the DP heuristics or establishing the LP optimality of the RMP at a given node of the B&P tree.

4.4.3 Comparison of Root-Node B&P Solution with Decomposition Heuristics

To complete our computational discussion, we compare the root-node B&P solutions against the results in both Ghoniem et al. (2013) and Solak et al. (2014), bearing in mind that these references used comparable hardware/software settings. For example, the column generation heuristic in Ghoniem et al. (2013) produced solutions within 4% optimal in over 900 CPU seconds for instances where $(|S|, |K|) = (10, 50)$. In contrast, using comparable instances, the root-node solutions of our B&P algorithm exhibited an optimality gap of 0.3% and were obtained in about 15 CPU seconds on average. This striking difference is entirely attributed to solving the pricing subproblem using our DP algorithms in lieu of CPLEX. As is typical of CG approaches, the solver often experienced numerical difficulties in solving the subproblem (spending excessive amounts of time to generate a particular new column) and experienced a long tailing-off effect that is largely circumvented using the proposed DP approaches.

Solak et al. (2014) focused on instances where $(|S|, |K|) = (25, 50)$ and found a Benders decomposition approach to yield better results over using CPLEX with a time limit of one CPU hour. However, the Benders decomposition approach produced solutions in nearly 2400 CPU seconds with an average optimality gap of about 8-9%. In contrast, over comparable instances, our B&P root node solutions exhibit an optimality gap of 2% on average and are obtained within about 160 CPU seconds.

Furthermore, the overall branch-and-price algorithm itself produced global optimal solutions in CPU times that are significantly shorter than those reported in the aforementioned two references. It is our conclusion that the proposed B&P algo-

Table 4.7: Comparative results for CPLEX vs. branch-and-price algorithm over 60 instances

		CPLEX (B&B/C Solver)			*B&P Algorithm*								
					Root-Node B&P				*Overall B&P Algorithm*				
$(\|\mathcal{S}\|, \|\mathcal{K}\|)$	*Instance*	Value*	Gap%	CPU(s)	LB	UB	Gap%	CPU(s)	Value*	Gap%	CPU (s)	Nodes	Savings%
(10, 20)	1	844	0	81.4	844	844	0	1.6	844	0	1.6	1	98.0
	2	818	0	2.5	818	818	0	0.5	818	0	0.5	1	80.6
	3	760	0	415.1	753	764	1.44	0.8	760	0	5.4	11	98.7
	4	919	0	277.1	919	919	0	1.3	919	0	1.3	1	99.5
	5	779	0	99.2	779	779	0	0.4	779	0	0.4	1	99.6
(10, 30)	1	1068	0	355.1	1067	1068	0.09	2.1	1068	0	4.2	3	98.8
	2	1106	0	17.3	1106	1106	0	1.1	1106	0	1.1	1	93.6
	3	981	0	1164.9	981	981	0	1.8	981	0	4.2	3	99.6
	4	1253	0	460.2	1239	1265	2.06	7.3	1253	0	385.1	275	16.3
	5	1284	0	3121.3	1284	1284	0.00	5.4	1284	0	5.4	1	99.8
(10, 40)	1	1462	4.4	3600.0	1462	1462	0	4.7	1462	0	4.7	1	99.9
	2	1465	0	2084.0	1460	1465	0.27	2.5	1465	0	9.3	13	99.6
	3	1587	5.9	3600.0	1585	1604	1.12	10.6	1587	0	20.4	5	99.4
	4	1381	6.4	3600.0	1370	1468	6.61	10.5	1381	0	110.8	55	96.9
	5	1475	7.1	3600.0	1475	1475	0	13.7	1475	0	13.7	1	99.6
(10, 50)	1	1828	0	2063.6	1823	1828	0.27	4.3	1828	0	9.1	3	99.6
	2	1581	15.7	3600.0	1539	1539	0	13.6	1539	0	13.6	1	99.6
	3	1791	11.7	3600.0	1789	1800	0.61	24.9	1789	0	107.7	5	97.0
	4	1726	8.8	3600.0	1726	1726	0	8.0	1726	0	8.0	1	99.8
	5	2029	14.6	3600.0	1985	1993	0.43	14.0	1993	0	397.7	193	89.0
(20, 20)	1	743	6.8	3600.0	742	743	0.13	3.6	743	0	12.8	3	99.6
	2	704	15.7	3600.0	704	704	0	5.9	704	0	5.9	1	99.8
	3	668	17.8	3600.0	668	668	0	36.8	668	0	36.8	1	99.0
	4	691	11.8	3600.0	691	691	0	50.0	691	0	50.0	1	98.6
	5	731	24.1	3600.0	730	731	0.14	1.3	731	0	3.6	9	99.9
(20, 30)	1	960	13.2	3600.0	951	955	0.42	20.4	955	0	104.1	7	97.1
	2	952	15.5	3600.0	939	969	3.10	8.8	952	0	821.1	383	77.2
	3	897	23.0	3600.0	897	897	0	4.6	897	0	7.8	3	99.8
	4	964	24.1	3600.0	931	955	2.41	10.7	941	0	277.6	81	92.3
	5	1072	32.6	3600.0	981	998	1.70	4.3	989	0	102.3	99	97.2
(20, 40)	1	1336	18.7	3600.0	1263	1264	0.08	20.9	1264	0	100.1	7	97.2
	2	1290	28.0	3600.0	1223	1229	0.49	10.2	1228	0	45.6	21	98.7
	3	1553	28.1	3600.0	1406	1425	1.33	15.4	1414	0	1365.8	303	62.1
	4	1427	25.7	3600.0	1350	1350	0.00	42.5	1350	0	42.5	1	98.8
	5	1274	22.5	3600.0	1223	1223	0.00	10.2	1223	0	10.2	1	99.7
(20, 50)	1	1880	40.7	3600.0	1504	1504	0.00	46.9	1504	0	46.9	1	98.7
	2	1795	45.0	3600.0	1330	1348	1.34	63.5	1344	0.35	3600.0	201	0.0
	3	1650	34.7	3600.0	1458	1475	1.08	28.1	1467	0	2006.5	409	44.3
	4	1610	30.1	3600.0	1449	1449	0.00	19.8	1449	0	19.8	1	99.4
	5	1600	31.6	3600.0	1502	1507	0.27	42.2	1505	0	53.9	5	98.5
(25, 20)	1	680	20.7	3600.0	680	680	0.00	21.5	680	0	21.5	1	99.4
	2	672	22.6	3600.0	668	668	0.00	38.2	668	0	38.2	1	98.9
	3	766	23.0	3600.0	764	766	0.26	57.5	766	0	148.1	5	95.9
	4	685	21.4	3600.0	678	684	0.88	4.8	683	0	38.6	5	98.9
	5	675	20.5	3600.0	675	675	0.00	1.8	675	0	1.8	1	100.0
(25, 30)	1	894	16.6	3600.0	894	894	0.00	35.7	894	0	101.2	3	97.2
	2	910	27.8	3600.0	896	900	0.33	7.1	897	0	975.4	21	72.9
	3	1149	43.8	3600.0	978	982	0.31	5.6	982	0	19.6	7	99.5
	4	971	20.2	3600.0	964	968	0.41	173.8	964	0	300.0	3	91.7
	5	1054	25.7	3600.0	1054	1054	0.00	19.0	1054	0	19.0	1	99.5
(25, 40)	1	1247	30.9	3600.0	1074	1074	0.00	16.5	1074	0	16.5	1	99.5
	2	1631	48.5	3600.0	1193	1199	0.50	31.9	1194	0	76.3	5	97.9
	3	1477	46.1	3600.0	1159	1162	0.17	725.3	1161	0	2993.5	3	16.8
	4	1461	32.2	3600.0	1247	1300	4.08	15.8	1252	0	61.7	9	98.3
	5	1303	30.5	3600.0	1181	1186	0.42	11.7	1186	0	115.9	47	96.8
(25, 50)	1	2106	50.2	3600.0	1466	1466	0.00	164.5	1466	0	164.5	1	95.4
	2	1611	42.2	3600.0	1313	1352	2.81	118.3	1352	2.53	3600.0	55	0.0
	3	1684	44.0	3600.0	1392	1417	1.76	86.2	1417	1.21	3600.0	97	0.0
	4	1952	52.9	3600.0	1394	1470	5.17	399.2	1470	4.89	3600.0	11	0.0
	5	1993	49.0	3600.0	1469	1472	0.14	38.4	1472	0	72.2	9	98.0

* The value reported corresponds to an optimal objective value or to an upper bound if an instance was not solved within 1 CPU hour

Table 4.8: Comparative results for the heuristic and exact DP algorithms

		Number of Subproblems Solved and Percentage Breakdown Across DPs							*Time (s) to Solve Subproblems and Percentage Breakdown Across DPs*						
$(\lvert S\rvert, \lvert K\rvert)$	#	Total	H1-R%	H1-NR%	H2-R%	H2-NR/EZ%	H2-NR/H%	Exact %	Total	H1-R%	H1-NR%	H2-R%	H2-NR/EZ%	H2-NR/H%	Exact%
(10, 20)	1	175	64	16	16	2	1	1	1.2	8.3	0.0	25.0	25.0	8.3	33.3
	2	255	54	18	18	5	4	2	0.1	0.0	0.0	100.0	0.0	0.0	0.0
	3	800	40	23	23	7	5	2	4.3	2.3	0.0	37.2	30.2	9.3	20.9
	4	183	62	15	15	4	3	1	1.1	0.0	0.0	27.3	36.4	9.1	27.3
	5	155	63	16	16	3	1	1	0.1	0.0	0.0	100.0	0.0	0.0	0.0
(10, 30)	1	622	49	24	23	2	2	$\approx 0^{\dagger}$	2.5	4.0	0.0	72.0	12.0	4.0	8.0
	2	242	66	16	16	1	1	≈ 0	0.3	0.0	0.0	100.0	0.0	0.0	0.0
	3	481	48	21	21	6	2	1	2.8	3.6	0.0	46.4	32.1	7.1	10.7
	4	22322	42	24	24	6	4	1	355.1	0.5	0.2	19.9	16.8	16.4	46.2
	5	331	61	18	17	3	2	1	4.3	2.3	0.0	25.6	16.3	11.6	44.2
(10, 40)	1	533	56	20	19	2	2	1	2.3	4.3	0.0	95.7	0.0	0.0	0.0
	2	1675	45	23	23	4	4	2	2.9	3.4	0.0	69.0	6.9	10.3	10.3
	3	837	53	20	20	3	3	1	15.3	0.7	0.0	19.6	5.2	9.2	65.4
	4	5019	34	24	24	8	7	3	86.8	0.8	0.2	24.0	14.3	20.9	39.9
	5	568	62	16	16	3	2	1	6.8	1.5	0.0	22.1	2.9	8.8	64.7
(10, 50)	1	1036	41	26	26	3	3	1	5	0.0	0.0	46.0	2.0	10.0	42.0
	2	696	66	16	16	2	1	≈ 0	5.7	3.5	0.0	73.7	3.5	14.0	5.3
	3	922	56	20	20	2	1	1	100.3	0.1	0.0	2.6	0.4	25.5	71.4
	4	557	64	17	17	1	1	≈ 0	3	6.7	0.0	93.3	0.0	0.0	0.0
	5	25664	46	23	22	4	4	1	313	0.5	0.2	11.2	1.9	20.0	66.2
(20, 20)	1	424	50	21	21	4	1	1	11.6	0.9	0.0	27.6	55.2	0.9	15.5
	2	335	52	19	19	6	2	1	4.5	2.2	0.0	22.2	28.9	2.2	44.4
	3	223	45	20	20	10	4	2	35.5	0.3	0.0	6.2	71.3	1.7	20.6
	4	220	59	17	17	5	1	≈ 0	49.3	0.0	0.0	4.7	64.1	0.2	31.0
	5	652	39	24	24	8	4	1	2.6	7.7	0.0	73.1	11.5	3.8	3.8
(20, 30)	1	723	51	20	20	5	2	1	99.1	0.7	0.1	11.5	58.7	0.7	28.3
	2	46995	39	25	24	9	3	1	698.9	1.3	0.4	32.0	37.8	4.5	24.1
	3	389	59	17	16	4	2	1	5.4	11.1	0.0	46.3	27.8	3.7	11.1
	4	5696	40	24	24	8	4	2	261.1	1.0	0.3	23.2	53.4	4.4	17.7
	5	6829	38	25	24	7	4	2	89.1	2.1	0.7	37.6	38.4	5.7	15.5
(20, 40)	1	1351	54	19	19	4	3	1	82.3	1.7	0.1	22.0	47.4	1.6	27.2
	2	1784	46	22	22	6	3	1	31.2	5.1	0.6	59.3	21.5	4.5	9.0
	3	37285	39	25	24	7	5	1	1134	0.8	0.2	23.7	28.2	6.6	40.5
	4	369	66	13	13	4	3	1	39.6	1.5	0.0	8.8	40.9	6.8	41.9
	5	461	71	13	13	2	0	≈ 0	4.7	31.9	0.0	66.0	2.1	0.0	0.0
(20, 50)	1	712	54	18	18	5	4	2	52.9	1.7	0.0	26.1	12.5	21.7	38.0
	2	27292	40	23	23	9	4	1	3437.8	0.5	0.1	7.4	25.3	10.3	56.3
	3	34163	38	24	24	8	4	2	1777.5	1.2	0.4	24.5	41.7	10.8	21.4
	4	600	60	16	16	5	3	1	13.8	13.8	0.0	55.8	14.5	7.2	8.7
	5	908	62	17	16	3	2	1	18.9	7.4	0.5	81.5	5.8	2.1	2.6
(25, 20)	1	346	64	16	15	3	1	1	19.8	1.0	0.0	12.1	60.6	0.5	25.8
	2	224	44	20	20	8	5	3	37.7	0.3	0.0	9.5	61.8	2.4	26.0
	3	497	38	24	24	9	4	2	147.3	0.2	0.1	6.0	67.1	0.9	25.7
	4	434	44	22	22	8	3	1	37.2	0.8	0.0	14.8	59.7	0.8	23.9
	5	163	67	14	14	2	1	1	1.2	8.3	0.0	66.7	16.7	0.0	8.3
(25, 30)	1	437	49	19	19	8	3	2	99.2	0.5	0.0	7.4	63.8	0.6	27.7
	2	2606	36	24	24	10	4	2	967.1	0.2	0.0	3.5	79.4	1.2	15.7
	3	635	48	22	22	4	3	1	17.1	2.3	0.6	42.7	36.3	4.1	14.0
	4	739	45	21	21	9	3	1	296.7	0.2	0.0	5.3	66.2	0.6	27.7
	5	320	55	19	19	5	2	1	17.5	2.3	0.0	38.3	39.4	2.3	17.7
(25, 40)	1	368	77	10	10	2	1	0	12.4	11.3	0.0	30.6	46.0	0.8	11.3
	2	742	57	18	18	4	2	1	66.6	4.7	0.2	29.7	38.6	2.1	24.8
	3	638	48	20	19	9	3	1	2988.4	0.1	0.0	0.9	55.9	0.5	42.6
	4	1052	50	20	20	5	3	1	50.3	3.6	0.2	41.2	30.4	4.6	20.1
	5	4740	40	23	23	8	4	1	81.1	5.7	0.5	49.1	28.7	6.7	9.4
(25, 50)	1	581	56	18	18	4	3	2	159.3	0.8	0.1	13.5	19.0	9.4	57.4
	2	5297	37	24	24	10	4	2	3567.5	0.2	0.0	2.7	34.5	7.2	55.4
	3	8559	37	24	24	10	4	1	3600	0.3	0.1	5.9	45.4	4.0	45.3
	4	1425	43	22	22	9	2	1	3600	0.1	0.0	1.9	46.5	1.2	54.7
	5	1404	59	17	16	4	2	1	41.3	11.4	0.2	66.6	12.3	3.6	5.8

†: Fraction rounded down to 0.

rithm offers an attractive solution methodology that outperforms not only CPLEX, but also certain heuristic decomposition approaches for this vehicle routing-allocation problem.

4.5. Conclusions

This chapter develops an effective branch-and-price algorithm for the vehicle routing with demand allocation problem (VRDAP), where the pricing subproblem is solved, exactly or heuristically, using a specialized labeling type, dynamic programming (DP) algorithm. The computational efficacy of this DP approach stems primarily from the inclusion of preprocessing routines that enhance the label extension scheme by iteratively eliminating dominated (partial) solutions. The proposed exact DP algorithm, and five proposed heuristic variants, significantly reduce the computational effort associated with the solution of the pricing subproblem (as opposed to solving the latter as an MIP with CPLEX). The resulting speed up enabled the implementation of a B&P algorithm that greatly outperformed the use of CPLEX over a test-bed of 60 problem instances, with up to 25 delivery sites and 50 customers.

In our computational study, the heuristic DP algorithms were sequentially triggered to solve the pricing subproblem and the exact DP was invoked only when all heuristics variants have failed to produce a column with a negative reduced cost. In our experience, the heuristic DP algorithms largely contributed to producing attractive columns at a moderate computational expense, whereas the more time-consuming exact DP algorithm was invoked to solve only 1% of the subproblems, often to achieve LP optimality at a given node of the B&P tree. The proposed DP-based B&P algorithm achieved a substantial 86% savings in CPU time on average over CPLEX. Moreover, the B&P algorithm solved 56/60 instances to optimality within a time limit of 1 CPU hour (compared to 12/60 instances solved to optimality by CPLEX). For the 4/60 instances that were not solved to optimality, the B&P algorithm produced solutions having an optimality gap of 2% on average.

We recommend for future investigation the exploration of variable neighborhood search heuristics to solve the pricing subproblem within a B&P algorithm or as a stand-alone heuristic approach for the VRDAP itself. It may be worthwhile to also explore the impact of allowing multiple vehicle tours to access the same delivery site (as opposed to having node-disjoint routes), time-windows for deliveries, or multiple depots to serve customers.

CHAPTER 5
CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

This dissertation contributes to the literature on routing problems in general and bi-level transportation, in particular. The first essay proposes an exact algorithm for the generalized vehicle routing problem (GVRP). The remainder of the dissertation investigates, heuristically in Chapter 3 and via an exact branch-and-price algorithm in Chapter 4, a Vehicle Routing with Demand Allocation Problem (VRDAP) that arises in food bank pallet distribution problem.

5.1. Summary of Findings

The first essay develops a branch-cut-and-price (BCP) algorithm for the GVRP. The problem is first formulated as a set-partitioning model in which capacity cuts were enforced in order to strengthen its underlying relaxation and the associated lower bound. The columns generation pricing subproblem was solved using a specialized dynamic programming algorithm that constitutes the kernel of our contribution. Further, an effective heuristic was developed with the purpose of generating high quality feasible solutions and, thus, relatively tight primal bounds for the BCP algorithm. The proposed BCP algorithm compares favorably against the state of the art exact algorithm for GVRP. Although it does not computationally outperform it for all benchmark instances in the literature, our algorithm provably solved to optimality 8 out of 9 open instances in the literature. Furthermore, for instances generated with random clustering (as opposed to proximity based clustering), the BCP algorithm performs significantly better than the branch-and-cut algorithm of Bektas et al. (2011).

Motivated by pallet distribution problems for food banks, the Second Essay proposes a multi-start optimization-based heuristic for the VRDAP. Central to the success of our methodology is the combined benefit of local search optimization-based routines that enable improvements in the routing as well as the customer assignment decisions, while avoiding more computationally taxing decomposition techniques in the spirit of column generation or Benders decomposition. Whereas these local search routines ensure intensification and solution refinement in certain areas of the feasible space, a perturbation mechanism is introduced in the algorithm in order to escape from local optima and to shift the search to diverse areas of the feasible space. For each of the 100 randomly generated instances in our testbed, three different objective weights were considered, placing equal or greater emphasis on either the vehicle routing cost or the customer travel cost. The proposed heuristic substantially outperforms the best incumbent solution identified by CPLEX within one CPU hour. It also greatly outperforms two decomposition heuristics in the literature in terms of solution quality and CPU time. At last, the impact of key components of the proposed heuristic on the performance of the algorithm is investigated in terms of the value of the heuristic solution accruing from its inclusion. This assessment of different algorithmic features can in our opinion guide the customization of solution methodologies for similar bi-level logistical and routing problems.

Essay Three develops a branch-and-price algorithm for the aforementioned food bank routing problem. The proposed column generation approach is grounded in solving the pricing subproblem using a labeling type dynamic programming (DP) algorithm. The specially-tailored DP algorithm, the manner in which a label tracks both routing and customer assignment decisions, and the devised dominance rules form our contribution and have enabled the algorithm to obtain the optimal solution of numerous instances in our testbed for this challenging problem. The computational efficiency of the proposed algorithm is further improved by developing five heuristic variants of the DP algorithm. The resulting speedup enables the implementation of a branch-and-price algorithm that greatly outperforms the use of CPLEX over a test-bed of 60 problem instances.

5.2. Directions for Future Research

We recommend for future research studying a new variant of the GVRP problem investigated in Essay One. In this variant, each node can only supply a fraction of the demand of its corresponding cluster. As such, routes may need to visit several nodes from each cluster. This new variant of GVRP has several potential applications. Especially, it can be used to optimize the routes that are used for picking up items that are dispersed among different locations of the a warehouse (which is the warehousing technique currently being used by the Prime Now service of Amazon).

As discussed in Essay Two, the VRAP problem lacks any local search heuristic that can explore a neighborhood by making simultaneous adjustments in routing and assignment decisions. As an extension for Essay Two, we recommend developing a specialized optimization based local search heuristic that enables us to tackle such difficulty. Such procedure revises the routing and assignment decisions by solving an integer linear programming model.

We would also like to study VRAP in the context of drone delivery where delivery sites are drone stations and the objective is to minimize the vehicle routing cost while assuring that customers are served by drone/vehicle during a specific time windows.

APPENDIX A
MIP FORMULATION FOR THE VRDAP

Using the notation introduced in Chapter 3, the VRDAP problem is modeled as a 0-1 MIP (Ghoniem et al. 2013) using the following decision variables:

- $s_{ik} \in \{0,1\}$: $s_{ik} = 1 \Leftrightarrow$ customer k is assigned to site i, $\forall i \in S, k \in K$.

- $e_{ij}^v \in \{0,1\}$: $e_{ij}^v = 1 \Leftrightarrow$ arc (i,j) is included in vehicle tour v, $\forall v \in V, (i,j) \in E$.

- $\theta_{ik}^v \in \{0,1\}$: $\theta_{ik}^v = 1 \Leftrightarrow$ site i is visited by vehicle tour v and customer k is assigned to site i, $\forall v \in V, i \in S, k \in K$.

- q_i^v: Total cumulative deliveries made upon serving site i in vehicle tour v, $\forall v \in V, i \in S$.

$$\text{Min} \sum_{v \in V} \sum_{(i,j) \in E} c_{ij}\, e_{ij}^v + \sum_{i \in S} \sum_{k \in K} f_{ik}\, s_{ik} \tag{A.1}$$

$$\text{subject to} \sum_{v \in V} \sum_{i \in N - \{j\}} e_{ij}^v \leq 1 \qquad \forall j \in S \tag{A.2}$$

$$\sum_{j \in S} e_{0j}^v \leq 1 \qquad \forall v \in V \tag{A.3}$$

$$\sum_{i \in N - \{j\}} e_{ij}^v - \sum_{i \in N - \{j\}} e_{ji}^v = 0 \qquad \forall v \in V, j \in N \tag{A.4}$$

$$q_j^v \geq q_i^v + \sum_{k \in K} d_k s_{jk} - 2Q(1 - e_{ij}^v) + Q e_{ji}^v \qquad \forall v \in V, i \neq j \in S \tag{A.5}$$

$$\theta_{ik}^v \geq s_{ik} + \sum_{j \in N - \{i\}} e_{ij}^v - 1 \qquad \forall v \in V, i \in S, k \in K \tag{A.6}$$

$$\theta_{ik}^v \leq \sum_{j \in N - \{i\}} e_{ij}^v \qquad \forall v \in V, i \in S, k \in K \tag{A.7}$$

$$\theta_{ik}^v \leq s_{ik} \qquad \forall v \in V, i \in S, k \in K \tag{A.8}$$

$$q_i^v \leq Q \sum_{j \in N - \{i\}} e_{ij}^v \qquad \forall v \in V, i \in S \tag{A.9}$$

$$\sum_{k \in K} d_k \theta_{ik}^v \leq q_i^v \qquad \forall v \in V, i \in S \tag{A.10}$$

$$\sum_{v \in V} \sum_{i \in S} \theta_{ik}^v = 1 \qquad \forall k \in K \tag{A.11}$$

$$e, s \text{ binary}, \theta, q \geq 0. \tag{A.12}$$

The objective function (A.1) minimizes the total routing and customer travel cost. Constraint (A.2) ensures that any delivery site is visited at most once. Constraint (A.3) requires any vehicle tour to have at most one arc leaving the central depot (node 0). Constraint (A.4) enforces flow balance for the central depot and any delivery site. Constraint (A.5) captures cumulative deliveries along a vehicle tour and serves as lifted Miller-Tucker-Zemlin subtour elimination constraints. Constraints (A.6)-(A.8) ensure that θ_{ik}^v assumes a value of 1 if and only if delivery site i is visited by vehicle v and customer k is assigned to delivery site i. Constraint (A.9) establishes an upper bound on the cumulative delivery made upon visiting any delivery site i. Constraint (A.10) requires the cumulative delivery made upon serving site i by the relevant vehicle tour to be at least the demand allocated to i itself. Constraint (A.11) ensures that every customer is assigned to exactly one delivery site and one vehicle tour. Constraint (A.12) introduces logical binary and nonnegativity restrictions on decision variables.

APPENDIX B
NOTATIONS

Summary of the notation used in Chapter 4.

- K^R: reduced set of customers after preprocessing procedure.
- S^R: reduced set of delivery sites after preprocessing procedure.
- Label $\mathcal{L}$: any partial solution of pricing subproblem is represented by a label such as $\mathcal{L}$.
- $P_\ell = (0, v_1, \ldots, v_\ell)$: an ordered set of delivery sites visited by $\mathcal{L}$.
- A_ℓ: an ordered set of clustered customers respectively associated with each delivery site of P_ℓ.
- C_ℓ: total cost of $\mathcal{L}$.
- Q_ℓ: total demand of all customers included in $\mathcal{L}$.
- σ_ℓ: set of sites visited by $\mathcal{L}$.
- χ_ℓ: set of customers served in $\mathcal{L}$.
- $\tilde{\sigma_\ell}$: Set of sites that if visited in any future extension of $\mathcal{L}$, would yield a dominated solution.
- $\tilde{\chi_\ell}$: set of customers that if included in any future extension of $\mathcal{L}$ would result in a dominated label.

- $K^{'}$: when extending label $\mathcal{L}$ to site v, $K^{'}$ is the set of customers that can be assigned to v. (initially, $K^{'} = K^{R} \setminus \chi_{\ell}$. Then, it is reduced using rules (1)-(3)).
- Ψ: when extending $\mathcal{L}$ to site v, Ψ is the subset of customers in $K^{'}$ that their closest delivery site is v.
- Ω: when extending $\mathcal{L}$ to site v, $\Omega \subseteq \Psi$ is the set of customers that must be assigned to v, otherwise, the label $\mathcal{L}$ will become a dominated label.

APPENDIX C

NUMERICAL EXAMPLE FOR THE DP ALGORITHM INTRODUCED IN CHAPTER 4

Using the example discussed in Chapter 4, we illustrate several steps of the DP algorithm (Algorithm 6) and procedure GET-LABELS. For the sake of brevity, we present the extension of only a few labels (among many labels) generated in each iteration. Recall that our example involves a set of delivery sites, $S = \{1, 2, \dots, 6\}$, and a set of customers, $K = \{a, b, \dots, k\}$. The assignment and routing cost matrices are given in Tables 4.1 and 4.2. Incorporating the dual values into the original cost matrices results in the updated matrices in Tables 4.5 and 4.6. The capacity of vehicle $Q = 8$ and all customers have unit demands. By preprocessing, these sets of sites and customers respectively reduce to $S^R = \{1, 2, 3, 4, 6\}$ and $K^R = \{a, b, c, d, e, f, g, k\}$. At last, each label is denoted by $\mathcal{L} = (P_\ell, A_\ell, C_\ell, Q_\ell)$ (see Appendix B for a summary of notations). Also, recall that procedure GET-LABELS$(\mathcal{L}, v_\ell)$ constructs all the customer subsets that can be feasibly assigned to v_ℓ. Many of these feasible customer subsets, however, would result in dominated labels. Procedure GET-LABELS uses three rules to detect such dominated subsets. For clarity in the exposition, these three rules are briefly reviewed:

- **Rule 1: Eliminating Unattractive Customers for** v_ℓ

  Consider $K^{'} = K^R \setminus \chi_\ell$ and $S^{'} = S^R \setminus (\sigma_\ell \cup \tilde{\sigma_\ell})$. All customers $k \in K^{'}$ such that $w_{k,v_\ell} \geq 0$ are removed from $K^{'}$, because their assignment to v_ℓ would worsen the solution. Further, for any customer $k \in K^{'}$, if there exists a site $v_i \in \sigma_\ell$ such that $w_{k,v_\ell} \geq w_{k,v_i}$, then customer k is removed from $K^{'}$.

- **Rule 2: Using Unassigned Customers for which v_ℓ is Closest**

  Let Ψ be the subset of customers in $K^{'}$ such that their closest delivery site in $S^{'}$, in the sense of the f-values, is v_ℓ. Observe that any customer in Ψ, if it is not assigned to v_ℓ itself, it will never be assigned to any future delivery site that is added to $\mathcal{L}$ after v_ℓ. Also, let Ω be the set of all customers $o \in \Psi$ for which there exists a customer k included in label $\mathcal{L}$ such that k is assigned to v_i, $d_k \geq d_o$, and $w_{k,v_i} \geq w_{o,v_\ell}$. If $o \in \Omega$ is not assigned to v_ℓ, then o will never be assigned to any label $\mathcal{L}^{'}$ that stems from $\mathcal{L}$. Furthermore, by removing k from site v_i and assigning o to v_ℓ instead, we can construct a better label. As a result, for the new label $\mathcal{L}^{'}$ to be non-dominated, assign all customers in Ω to v_ℓ and remove all the members of Ω from $K^{'}$ and Ψ. Note, of course, that if the inclusion of Ω was not feasible due to the tour capacity, this label would be dominated and should not be given further consideration.

- **Rule 3: Identifying So-Called Inferior Customers**

  A customer $k \in K^{'}$ is deemed to be *inferior* to $o \in \Psi$, if $d_k \geq d_o$ and $w_{k,v_\ell} \geq w_{o,v_\ell}$. If customer o is not assigned to v_ℓ, then none of its inferior customers can be assigned to v_ℓ (otherwise, this would result in a dominated label).

Algorithm 6 starts at the depot by creating an initial label $\mathcal{L}_0 = \{((0), (\varnothing), 0, 0)\}$ and setting $\Gamma = \{0\}$ (lines 2-3). The while loop removes the depot from Γ and extends its only label to delivery sites of S^R. Starting from delivery site 1, procedure GET-LABELS$(\mathcal{L}_0, 1)$ constructs all the non-dominated labels that can be obtained by extending label $\mathcal{L}_0$ to delivery site 1 and assigning to it a feasible subset of customers in K^R. According to Rule 1, assigning all the customers $\{e, f, g, k\}$ to site 1 would result in a dominated label, thus, $K^{'} = \{a, b, c, d\}$. In Rule 2, $\Psi = \{a\}$ and $\Omega = \varnothing$. Rule 3 detects all the customers $\{b, c, d\}$ inferior to $a \in \Psi$. In other words, if a is not assigned to site 1, no other customer can be assigned to it as well. Therefore, GET-LABELS($\mathcal{L}_0, 1$) considers the following subsets of customers: $\{a\}, \{a,b\}, \{a,c\}, \{a,d\}, \{a,b,c\}, \{a,b,d\}, \{a,c,d\}$ and $\{a,b,c,d\}$. Total of 8 new labels are generated, each having one of these subsets assigned to site 1. Since delivery site 1 has some new labels that are not extended yet, $\Gamma = \{1\}$.

Similarly, several new labels are generated for delivery site 2. Rule 1 reduces K^R to $K^{'} = \{a,b,c,d,f\}$. According to Rule 2, $\Psi = \{b,d\}, \Omega = \varnothing$. Rule 3 detects customer f to be inferior to both b and d. Also, c and d are both inferior to b. Therefore, 13 labels are generated by assigning each of the following subsets to site 2: $\{a\}, \{b\}, \{a,b\}$, $\{b,c\}, \{b,d\}, \{a,b,c\}, \{a,b,d\}$, $\{b,c,d\}, \{b,d,f\}, \{a,b,c,d\}$, $\{a,b,d,f\}, \{b,c,d,f\}$ and $\{a,b,c,d,f\}$. Due to the introduction of new labels for site 2, $\Gamma = \{1,2\}$. The for loop (line 6) is repeated for sites 3, 4 and 6 and creates several new labels for all of them. After the first iteration of while loop, $\Gamma = \{1,2,3,4,6\}$.

The second iteration of while loop starts by selecting delivery site 1 from Γ and extending its labels that are not extended before. For the sake of brevity, here we do the extensions for only 2 out of 8 labels generated in the previous iteration.

For label $\mathcal{L}_1 = ((0,1), (\varnothing, \{a,d\}), -50, 2)$, $\sigma_{\ell_1} = \{0,1\}$ and $\bar{\sigma}_{\ell_1} = \{2,3,4\}$. Note that, extending $\mathcal{L}_1$ to any delivery site in $\bar{\sigma}_{\ell_1}$ would result in a dominated label (removing customer d from site 1 and assigning it to the visited site from $\bar{\sigma}_{\ell_1}$ creates a better label). Therefore, $\mathcal{L}_1$ is only extended to delivery site 6 by calling procedure GET-LABELS($\mathcal{L}_1, 6$). Rule 1, rules out all the remaining customers except k. Only one new label $((0,1,6), (\varnothing, \{a,d\}, \{k\}), -27.5, 3)$ is generated.

The same happens when extending $\mathcal{L}_2 = ((0,1), (\varnothing, \{a,c\}), -60, 2)$ to delivery site 6. $\sigma_{\ell_2} = \{0,1\}$ and $\bar{\sigma}_{\ell_2} = \{2,3,4\}$. The only label generated by GET-LABELS($\mathcal{L}_2, 6$) is $((0,1,6), (\varnothing, \{a,c\}, \{k\}), -37.5, 3)$. Following the same steps, other 6 labels are extended to other delivery sites, generating several new labels. In the next iterations of while loop, the non-extended labels of other members of Γ are extended as above.

As a final example, we consider an iteration of while loop where label $\mathcal{L}_3 = ((0,2,3),(\varnothing,\{a,b\},\{c\}),-92.5,3)$. For this label, $\sigma_{\ell_3} = \{0,2,3\}$ and $\bar{\sigma}_{\ell_3} = \{1\}$. Therefore, $\mathcal{L}_3$ is only extended to sites 4 and 6. For the delivery site 4, Rule 1 detects assignment of customers k,d,e to site 4 a dominated decision. Among the remaining customers, i.e. f and g, Rule 3 detects g to be inferior to f. Therefore, only the following two new labels are generated: $((0,2,3,4),(\varnothing,\{a,b\},\{c\},\{f\}),-118,4)$ and $((0,2,3,4),(\varnothing,\{a,b\},\{c\},\{g,f\}),-138,5)$. Similarly, the procedure GET-LABELS$(\mathcal{L}_3,6)$ only considers the assignment of customer k as a non-dominated decision. Therefore, only one label $((0,2,3,6),(\varnothing,\{a,b\},\{c\},\{k\}),-60.5,4)$ is generated by this procedure.

BIBLIOGRAPHY

Akinc U, Kizhanatham S (1992) Optimal routing and process scheduling for a mobile service facility. *Networks* 22(2): 163–183.

Baldacci R, Dell'Amico M, Salazar González J. (2007). The capacitated m-ring-star problem. *Operations Research*, 55(6): 1147–1162.

Baldacci R, Dell'Amico M, González JS (2007) The capacitated m-ring-star problem. *Operations Research* 55(6): 1147–1162.

Baldacci R, Christofides N, Mingozzi A (2008) An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming* 115(2): 351–385.

Baldacci R, Bartolini E, Laporte G (2010) Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society* 61(7): 1072–1077.

Baldacci R, Mingozzi A, Roberti R (2011) New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* 59(5): 1269–1283.

Baltz A, El Ouali M, Jağer G, Sauerland V, Srivastav A (2015) Exact and heuristic algorithms for the travelling salesman problem with multiple time windows and hotel selection. *Journal of the Operational Research Society* 66(4): 615–626.

Beasley JE, Nascimento EM (1996). The vehicle routing-allocation problem: A unifying framework. *Top*, 4(1): 65–86.

Bektas T, Erdogan G, Røpke S (2011) Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science* 45(3): 299–316.

Boland N, Dethridge J, Dumitrescu I (2006) Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters* 34(1): 58–68.

Current, Schilling (1992). The median tour and maximal covering tour problems: Formulations and heuristics, *European Journal of Operational Research*, 73(1): 114–126.

Christofides N, Mingozzi A, Toth P (1981) Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming* 20(1): 255–282.

de Souza, LV, Siqueira PH (2010) Heuristic methods applied to the optimization school bus transportation routes: a real case, in *Trends in Applied Intelligent Systems* 247–256, Springer.

De Franceschi R, Fischetti M and Toth P (2006) A new ilp-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3): 471–499.

Dell'Amico M, Righini G, Salani M (2006) A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science.* 40(2): 235–247.

Dror M (1994) Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research* 42(5): 977–978.

Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(2): 342–354.

Feeding America (2015). *Feeding Families, Feeding Hope – 2015 Annual Report.* www.feedingamerica.org.

Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3): 216–229.

Fischetti M, Salazar González J, Toth P (1997) A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* 45(3): 378–394.

Fukasawa, R, Longo H, Lysgaard J, de Aragão M P, Reis M, Uchoa E, Werneck R F (2006) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming* 106(3): 491–511.

Gendreau M, Laporte G, Semet F (1997) The covering tour problem. *Operations Research* 45(4): 568–576.

Ghiani G, Improta G (2000) An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research* 122(1): 11–17.

Ghoniem A and Sherali HD (2009). Complementary column generation and bounding approaches for set partitioning formulations. *Optimization Letters*, 3(1), 123–136.

Ghoniem A, Scherrer C and Solak S (2013). A specialized column generation approach for a vehicle routing problem with demand allocation. *Journal of the Operational Research Society*, 64 (1): 114–124.

Ghoniem A, Farhadi F, Reihaneh M (2015) An accelerated branch-and-price algorithm for multiple-runway aircraft sequencing problems. *European Journal of Operational Research* 246(1): 34–43.

Gohari S, Salmasi N (2015) Flexible flowline scheduling problem with constraints for the beginning and terminating time of processing of jobs at stages. *International Journal of Computer Integrated Manufacturing* 28(10): 1092–1105.

Hooker JN, Ottosson G (2003) Logic-based Benders decomposition. *Mathematical Programming* 96, 33–60.

Hoshino EA, De Souza, CC (2012) A branch-and-cut-and-price approach for the capacitated m-ring–star problem. *Discrete Applied Mathematics* 160(18): 2728–2741.

Irnich S, Villeneuve D (2006) The shortest-path problem with resource constraints and k-cycle elimination for $k = 3$. *INFORMS Journal on Computing* 18(3): 391–406.

Karapetyan D, Gutin G (2012) Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. *European Journal of Operational Research* 219(2): 234–251.

Kim B-I, Kim S, Sahoo S (2006) Waste collection vehicle routing problem with time windows. *Computers & Operations Research* 33(12): 3624–3642.

Labbé M and Laporte G (1986). Maximizing user convenience and postal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26(2): 21–35.

Labbé M, Laporte G, Martin IR and Gonzalez JJS (2004). The ring star problem: polyhedral analysis and exact algorithm. *Networks*, 43: 177–189.

Labbé M, Laporte G, Rodríguez Martín I, Salazar González JJ (2005) Locating median cycles in networks. *European Journal of Operational Research* 160(2): 457–470.

Lysgaard J, Letchford A, Eglese R (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming* 100(2): 423–445.

Martinelli R, Pecin D, Poggi M (2014) Efficient elementary and restricted non-elementary route pricing. *European Journal of Operational Research* 239(1): 102–111.

Murty KG and Djang PA (1999). The us army national guard's mobile training simulators location and routing problem. *Operations Research*, 47(2): 175–182.

Nagy G and Salhi S (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research* 177(2): 649–672.

Naji-Azimi Z, Salari M, Toth P (2010). A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research*, 207 (3): 1227–1234.

Naji-Azimi Z, Salari M, Toth P (2012). An integer linear programming based heuristic for the capacitated m-ring-star problem. *European Journal of Operational Research*, 217 (1): 17–25.

Peréz JAM, Moreno-Vega JM and Martin IR (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151: 365–378.

Reihaneh M, Karapetyan D (2012) An Efficient Hybrid Ant Colony System for the Generalized Traveling Salesman Problem. *Algorithmic Operations Research* 7(1): 22–29.

Reihaneh M, Ghoniem A (2017) A multi-start optimization-based heuristic for a food bank distribution problem, *Journal of the Operational Research Society*, forthcoming.

Reihaneh M, Ghoniem A (2018) A branch-cut-and-price algorithm for the generalized vehicle routing problem, *Journal of the Operational Research Society* 69(2): 307–318.

Renaud J, Boctor FF and Laporte G (2004). Efficient heuristics for median cycle problems. *Journal of the Operational Research Society*, 55: 179–186.

Renaud J, Boctor F (1998) An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operational Research* 108(3): 571–584.

Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks* 51(3): 155–170.

Salari M, Reihaneh M, Sabbagh, M.S (2015) Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Computers & Industrial Engineering* 83 : 244–251.

Savelsbergh M (1997). A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45: 831-841.

Schittekat P, Kinable J, Sorensen K, Sevaux M, Spieksma F, Springael J (2013) A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research* 229(2): 518–528.

Schneider M, Stenger A, Hof J (2015). An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *OR Spectrum* 37(2):353–387.

Schneider M, Stenger A, Goeke D (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science* 48(4):500–520.

Sherali HD, Smith JC (2001) Improving discrete model representations via symmetry considerations. Management Science. 47(10): 1396–1407.

Solak S, Scherrer C and Ghoniem A (2014). The stop-and-drop problem in nonprofit food distribution networks. *Annals of Operations Research*, 221(1): 407–426.

Soubbotina TP (2004). *Beyond Economic Growth: An Introduction to Sustainable Development.* Second Edition, The World Bank, Washington, D.C.

de Souza LV and Siqueira PH (2010). Heuristic methods applied to the optimization school bus transportation routes: a real case. In *Trends in Applied Intelligent Systems*, Springer, pp. 247–256.

Tarantilis CD, Zachariadis EE, Kiranoudis CT (2008) A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing*, 20(1):154–168.

Uvin P (1994). The state of world hunger. *Nutrition Reviews*, 52(5): 151-161.

Vogt L, Poojari CA, Beasley JE (2007) A tabu search algorithm for the single vehicle routing allocation problem, Journal of the Operational Research Society, 58, 467–480.

Wolsey LA (1998) *Integer Programming.* Wiley, New York.

World Hunger (2015). Hunger in America: 2015 United States and Poverty Facts, www.worldhunger.org.

Yazdani M, Gohari S, Naderi B (2015) Multi-factory parallel machine problems: Improved mathematical models and artificial bee colony algorithm. *Computers & Industrial Engineering 81: 36–45.*