Department of Mathematics and Statistics

A Hybrid Method for Capacitated Vehicle Routing Problem

Mamon Radiy

This thesis is presented for the Degree of

Doctor of Philosophy

of

Curtin University of Technology

March 2010

To My Mother

Declaration

To the best of my knowledge and belief this thesis contains no materials previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Mamon Radiy

Abstract

The vehicle routing problem (VRP) is to service a number of customers with a fleet of vehicles. The VRP is an important problem in the fields of transportation, distribution and logistics. Typically the VRP deals with the delivery of some commodities from a depot to a number of customer locations with given demands. The problem frequently arises in many diverse physical distribution situations. For example bus routing, preventive maintenance inspection tours, salesmen routing and the delivery of any commodity such as mail, food or newspapers.

We focus on the Symmetric Capacitated Vehicle Routing Problem (CVRP) with a single commodity and one depot. The restrictions are capacity and cost or distance. For large instances, exact computational algorithms for solving the CVRP require considerable CPU time. Indeed, there are no guarantees that the optimal tours will be found within a reasonable CPU time. Hence, using heuristics and meta-heuristics algorithms may be the only approach. For a large CVRP one may have to balance computational time to solve the problem and the accuracy of the obtained solution when choosing the solving technique.

This thesis proposes an effective hybrid approach that combines domain reduction with: a greedy search algorithm; the Clarke and Wright algorithm; a simulating annealing algorithm; and a branch and cut method to solve the capacitated vehicle routing problem. The hybrid approach is applied to solve 14 benchmark CVRP instances. The results show that domain reduction can improve the classical Clarke and Wright algorithm by 8% and cut the computational time taken by approximately 50% when combined with branch and cut.

Our work in this thesis is organized into 6 chapters. Chapter 1 provides an introduction and general concepts, notation and terminology and a summary of our work. In Chapter 2 we detail a literature review on the CVRP. Some heuristics and exact methods used to solve the problem are discussed. Also, this Chapter describes the constraint programming (CP) technique, some examples of domain reduction, advantages and disadvantage of using CP alone, and the importance of combining

CP with MILP exact methods. Chapter 3 provides a simple greedy search algorithm and the results obtained by applying the algorithm to solve ten VRP instances. In Chapter 4 we incorporate domain reduction with the developed heuristic. The greedy algorithm with a restriction on each route combined with domain reduction is applied to solve the ten VRP instances. The obtained results show that the domain reduction improves the solution by an average of 24%. Also, the Chapter shows that the classical Clarke and Wright algorithm could be improve by 8% when combined with domain reduction. Chapter 4 combines domain reduction with a simulating annealing algorithm. In Chapter 4 we use the combination of domain reduction with the greedy algorithm, Clarke and Wright algorithm, and simulating annealing algorithm to solve 4 large CVRP instances. Chapter 5 incorporates the Branch and Cut method with domain reduction. The hybrid approach is applied to solve the 10 CVRP instances that we used in Chapter 4. This Chapter shows that the hybrid approach reduces the CPU time taken to solve the 10 benchmark instances by approximately 50%. Chapter 6 concludes the thesis and provides some ideas for future work. An appendix of the 10 literature problems and generated instances will be provided followed by bibliography.

Acknowledgement

My sincerest gratitude to my thesis supervisor Professor Lou Caccetta, for his invaluable guidance, helpful advices, and encouragement during the preparation of this thesis, as his guidance and kind help provide me with enough knowledge to understand the problem and develop solution algorithms . Also I would like to thank my associate supervisor Stephen Hill for his help.

To my collogue Mark Grigoleit my gratitude and special thanks, for his help and support. Also, my gratitude to the post graduate coordinator Professor Y. Hong Wu, for his kind assistance and sincere support for all post graduate students in general and me in particular. I would like to thank the staff of the department and post graduate students for their encouragement and support.

To my mother Fadia, my brother Anwar, my father Hamid and my wife Hala, my gratitude and love for their support and understanding. Also to my uncle Fawzi my deepest gratitude, for his encouragement and for his helpful advice in how to use x-press package.

Financially I'm grateful to Professor Lou Caccetta for his kind assistance and support. Also, I would like to thank Curtin University for providing me with a Curtin postgraduate award (CUPSA).

.

Table of Contents

Chapter 1

Introduction 1

1.1 Notation and Terminology 3

1.2 Problem Formulation... 8

1.3 Review and Summary of Thesis 9

Chapter 2

Literature review.. 13

2.1 Classical Heuristics 17

_2.1.1 The Clarke and Wright Algorithm (1964).. 17

_2.1.2 Sweep Algorithm (Wren and Holliday (1972)) 20

2.2 Metaheuristics.. 21

_2.2.1 Simulating Annealing (SA).. 21

_2.2.2 Genetic algorithms (GA) 22

2.3 Branch and Bound... 23

2.4 Cutting Plane Technique 26

2.5 Application of Branch and Cut Method to VRP.................................... 28

_2.5.1 The Laporte, Nobert and Desrochers (1985).. 28

_2.5.2 Achuthan, Caccetta and Hill (2003) Improved Branch and Cut Algorithm. 33

2.6 The Advantages and Disadvantages of Branch and Cut 35

2.7 Constraint Programming (CP) 36

_2.7.1 Binarization of Constraints 37

_2.7.2 Systematic Search Algorithms 38

_2.7.3 Consistency Techniques 38

_2.7.4 Constraint Propagation 39

_2.7.5 Value and Variable Ordering 40

_2.7.6 Reducing Search 40

2.8 Constraint Programming and Operations Research. 41

2.9 Integrating CP and OR Techniques 43

2.10 Constraint Programming and VRP 44

2.11 Advantages and Disadvantages of Integrating CP with OR 45

Chapter 3

Heuristics and Domain Reduction 46

3.1 A Simple Heuristic Algorithm for the Symmetric VRP 47

3.2 Calculations 55

3.3 Conclusion 61

Chapter 4

Heuristic Algorithm for CVRP 62

4.1 Domain Reduction 63

4.1.1 Computations 63

4.1.2 Distance Constrained Vehicle Routing Problem (DCVRP) 64

4.2 Clarke and Wright (C&W) Algorithm 68

4.3 Simulating Annealing Algorithm (SA) 69

4.4 Heuristics and large instances 71

4.5 Conclusion 72

Chapter 5

A hybrid Method to solve VRP 73

5.1 Domain Reduction condition and Implementation 73

5.2 Calculation 74

5.3 Conclusions 81

Chapter 6

Conclusion and Future Work 82

6.1 Future Work 83

Appendix A 84

Bibliography 96

Chapter 1

Introduction

Procurement, production and distribution are the traditional three stages for the supply chain. Managing the flow of materials and information inside and outside the production facilities has received increased attention over recent years. Furthermore, transporting goods and commodities contribute 20%-30% of the overall cost of the supply chain. Moving towards more complicated logistics options, transportation optimization has become an important factor in reducing the product cost.

Transporting raw materials to factories or goods to customers are the key objectives of a distribution network. Surveys done in 2001 by the Council of Logistics Management (CLM) in North America* showed that transportation represents 6 percent of the U.S. gross domestic product expenses.

The vehicle routing problem (VRP) is an important problem in the distribution network and has a significant role in reducing the cost and improving the service.

The problem is one of visiting a set of customers using a fleet of vehicles, respecting constraints on the vehicles, customers, drivers, and so on. The goal is to produce a minimum cost routing plan specifying for each vehicle, the order of the customer visits they make. The problem of vehicle scheduling was first formulated by Dantzig and Ramser (1959) and may be stated as a set of customers, each with a known location and a known requirement for some commodity, is to be supplied from a single depot by delivery vehicles, subject to the following conditions and constraints:

(a) The demands of all customers must be met.

(b) Each customer is served by only one vehicle.

(c) The capacity of the vehicles may not be violated (for each route the total demands must not exceed the vehicle capacity).

* AllBusiness.com (2007).

The objective of a solution may be stated in general terms as that of minimizing the total cost of delivery, namely the costs associated with the fleet size and the cost of completing the delivery routes (Christofides and Eilon (1969)).The problem frequently arises in many diverse physical distribution situations. For example bus routing, preventive maintenance inspection tours, salesmen routing and the delivery of any commodity such as mail, food or newspapers (Achuthan et al (1996)). The vehicle routing problem is an integer programming problem that falls into the category of **NP-Hard** problems. As the problems become larger, there will be no guarantee that optimal tours will be found within reasonable computing time (Achuthan et al (1991)).

Over the past 50 years vehicle routing or dispatching problems have been extensively studied by researchers around the word. Algorithms have been developed to improve both exact and heuristic methods. The major focus of this thesis is the development and implementation of a hybrid approach that combines **domain reduction** with heuristics and the branch and cut method. In this thesis we consider the capacitated vehicle routing problem (CVRP) where the problem is to determine delivery routes, one for each vehicle, which minimize the total distance traveled by all vehicles. Note that if the vehicle has infinite capacity, the CVRP may be viewed then as a symmetric traveling salesman problem (STSP). Much of the computational work on the CVRP has been motivated by the success of methods to solve the Travelling Salesman Problem (TSP). Branch and Cut is a method that has been used to solve larger STSP effectively, the method has also proven to be effective when used to solve larger CVRP.

The branch and cut method can be considered as an extension of branch and bound. As in the branch and bound method, one must compute a lower bound and an upper bound on a problem (minimizing problem) and divide the feasible region of a problem to create smaller sub-problems. The branch and bound finds a lower bound and upper bound at the start. If the two bounds are the same, then an optimal solution has been found. Otherwise, the feasible region is divided into sub-problems (branching). Note that, solving these subproblems will be easier than dealing with the original problem. At each stage a sub-problem is selected and an effort is made to find its optimal solution. An optimal solution is found for the problem when no more branching is possible.

The term Branch and Cut was coined by Padberg and Rinaldi (1987). The branch and cut solves the linear problem ignoring the integer constraints. After solving the problem without the integer constraints, the algorithm then generates a cut, if this cut is violated by the current solution then the generated cut inequality will be added as an extra constraint to the original problem. The process of solving the relaxation problem and generating the cuts is repeated until either an integer solution is found or until no more cutting planes are found. So in this case the problem splits into two sub-problems, the first with a constraint that is greater than or equal to the greatest integer in the intermediate result, and the second with a constraint less than or equal to the lesser integer. The process is repeated starting from solving the relaxed problem using the simplex method. However, in some NP-hard problems like the VRP the branch and cut method can take a long time to solve the problem and in some cases it fails to produce an optimal solution mainly because of the problem size. At this point using constraint programming (CP) may be helpful since CP is mainly developed to provide feasible solutions for different types of problems especially the large ones while branch and cut method showed the importance of using it to get the optimal solution for various **NP-hard** problems.

NP-hard problems are a true challenge and often attracted attentions for their importance in minimizing the cost or maximizing productivity. The approaches to solve the optimization problems and some needed notations and terminologies are discussed below.

1.1 Notation and Terminology

In the application of mathematical techniques to problems arising in science and technology, the problem that often arises is that of optimizing a function subject to a set of constraints. Usually the function to be optimized represents profit or cost, while the constraints reflect restrictions imposed by limited resources such as raw materials, market requirements, equipment availability, capacity and other restrictions. The problem may be expressed as:

Problem (1.1):

$$\text{minimize } Z = cx$$

subject to

$$Ax \leq b$$
$$x \geq 0$$

The problem is called a Linear Program (LP), when the objective function and constraint set are linear and called a Mixed Integer Linear Program (MILP), if some of the variables are specified as integer. The problem is a pure Integer Linear Program (ILP), if all variable values must be integral. The VRP can be formulated as either a MILP or ILP. Non-linear constraints problems or objectives are not considered in this thesis.

LP problems are easier to solve than both MILP and ILP problems. Since solving MILP or ILP problems normally requires the solution of one or more easier LP sub-problems, by dropping the integer restrictions or some of the other constraints. More formally, a problem (F) is a **relaxation** of a minimization problem (P) if:

- The set of feasible solutions of P is a subset of the feasible solution of F.
- The objective function of F bounds the objective function of P from below over the domain of F.

Solutions of the relaxations are used in a search tree technique, such as the method of Branch and Bound, or Branch and Cut, to obtain optimality. The sub-problem is said to be **fathomed**, if the objective function value of the optimal solution to the sub-problem is at least equal to the objective function value of the best known solution of the original problem.

The difficulty of a decision problem is classified into three classes: P, NP and NP-Complete. Problems for which polynomial time algorithms are known belong to the class P. In addition, an algorithm solves all instances of a problem by using a maximum number of steps that increases polynomially with the problem size. The problems which can be solved by a non-deterministic algorithm in polynomial time and all the problems in P belong to NP class. The class **NP-Complete** is a subset of NP having the property that all problems in NP can be reduced in polynomial time to one of them.

A problem is **NP-Hard** if every problem in NP is **polynomially** reducible to it. Usually MILP and ILP problems are NP-Hard. In the majority of cases, only exponential time algorithms are known for MILPs and ILPs. For this reason there is no assurance of finding the solution in a reasonable amount of time.

The following terms are used in the description of the solution space of a discrete optimization problem. We begin by considering the set of all possible solutions of a MILP or ILP. The restrictions to find the solutions may be described by a set of linear constraints, and the problem expressed in the form of Problem (1.1). Finding these constraints and their properties is the subject of polyhedral theory**.** A detail treatment of this subject is presented in the excellent book of Nemhauser and Wolsey (1988). Some basic aspects are briefly described below.

Given $S \subseteq R^n$, a point $x \in R^n$ is a convex combination of points of S if there exists a finite set of points $\{x_i\}_{i=1}^t$ in S and a vector $\lambda \in R^t$ of non-negative values with $\sum_{i=}^{t} \lambda_i = 1$ and $x = \sum_{i=1}^{t} \lambda_i x_i$. The **convex hull** of S, denoted by conv(S), is the set of all points that are convex combinations of S. Note that as a result finite S, conv(S) can be described by a finite set of linear inequalities. In addition $\min \{cx : x \in S\} = \min \{cx : x \in \text{conv}(S)\}$. Thus any MILP or ILP can be represented as an LP provided we know a set of linear inequalities that represent the solution space. Note that such a system of inequalities is usually incredibly large in number and generally unknown. To overcome these problems, the approach is to use a subset of the constraints defining conv(S) and/or constraints which are redundant in a minimal representation.

The inequality $\pi x \leq \pi_0$ is called a **valid inequality** for Problem (1.1) if it is satisfied for all points in *P*. A linear constraint that does not exclude any integer feasible points is called a **cutting plane**. If $\pi x \leq \pi_0$ is a valid inequality for *P*, and F = $\{x \in P: \pi x = \pi_0\}$, then F is called a **face** of *P*. A face of *P* is a **facet** of *P* if dim(F) = dim(*P*)-1. This leads to the result that for each facet F of *P*, one of the inequalities representing F is necessary in the description of *P*. Thus the use of facets in the description of the solution space yields a system of inequalities of smallest number. Also, if *P* defines the **convex hull** of integer solutions of a discrete optimization problem, then the use of facet defining inequalities is most likely to give the tightest lower bounds in a Branch and Cut scheme.

The VRP feasible and partial solutions may be modelled using a graph. A graph G is an ordered triple $(V(G), E(G), \Phi_G)$ consisting of a nonempty set V(G) of vertices, a set E(G) of edges disjoint from V(G) and an incidence function Φ_G that associates with each edge of G an unordered pair of vertices of G. If u and v are vertices of the graph *G* identified with an edge e, then e is incident with u and v*;* u and v are the ends of edge e. If each edge e = uv has a positive edge weight c_{uv} associated with it, then the graph is weighted. Consider the MILP formulation of CVRP with variables x = (x_{ij}). We can associate a weighted graph G with any solution $C = (x_{ij})$ of the problem as follows. V(G) = {0,1,.. . ,n}, E(G) = $\{(i, j) : x_{ij} > 0\}$, and the weight of edge (i, j) is c_{ij}.

The degree of a vertex u in a graph G is the number of edges of G incident with u. For a weighted graph G, the degree of vertex u refers to the sum total of edge weights, c_{ij} of edges incident with u. Arc set A(G) is used in place of E(G), if Φ_G specifies the vertices are ordered in its association.

A graph H = $(V(H), E(H), \Phi_H)$ is a sub-graph of G = (V(G},E(G), Φ_G) if $V(H) \subseteq V(G), E(H) \subseteq E(G), \Phi_H$ is the restriction of Φ_G to E(H). Let V' be a non-empty subset of V(G). A graph G[V'] whose vertex set is V' and whose edge set is the set of those edges of G that have both ends in V' is called an induced sub-graph of G. ε(G[V']) denotes the number of edges of G[V'].

A path in a graph G is a finite, non-empty alternating sequence W = $v_0, e_1, v_1, e_2, \ldots, e_n, v_n$ of vertices and edges, such that for $1 \leq i \leq n$, the ends of edge e_i are v_{i-1} and v_i. If the path has distinct vertices then it is a simple path. A cycle is a simple path with the origin v_0 and terminus v_n the same. A 2-cycle is a cycle on 2 vertices and is of the form W= v_0, a_1, v_1, a_2, v_0 where a_1 and a_2 are arcs from v_0 to v_1 and from v_1 to v_0, respectively.

A graph G is connected if there is a path between every pair of vertices; otherwise it is disconnected. A tree is a connected graph without cycles. A maximal connected sub-graph is called a component.

For a graph with n vertices, a **Hamiltonian cycle** is a cycle that visits each vertex exactly once and finishing the cycle at the starting vertex. The **Travelling Salesman Problem (TSP)** is to find a cycle through the n vertices that minimize the sum of the associated edge costs. Hence, any solution for TSP can be seen as a spanning Hamiltonian cycle of a minimum weight. Including a depot in the vertex set and considering more than one salesman results in a **Multiple Travelling Salesman Problem** that finds m cycles with a common vertex (representing the depot) which minimizes the sum of the associated edge costs. Note that the degree of the depot must be 2m and every other vertex has degree 2.

The **Bin Packing Problem (BPP)** is to assign each of the items to one of the m bins so that the number of bins used is minimized, with the sum of the weights of items in any particular bin at most c, where c is the common capacity. Note that vehicle routing problem (VRP) can be seen as a combination of TSP and BPP. Also, any solution to VRP with m vehicles can be viewed as m Hamiltonian cycles

Constraint satisfaction problems normally consist of finite variables with finite domains and finite constraints restricting the values of the variables. The problem solution will involve the use of logic to assign the variables with values from the domain so that all constraints are satisfied. The **Constraint Programming (CP)** method is the embedding of constraints in a logic programming language to solve constraint satisfaction problems. The method is based on the idea of using logic to satisfy a large number of constraints. The **Domain reduction** technique is one of the approaches to deal with constraint satisfaction problems. As the name suggest, the domain reduction technique is to use logic to reduce the domain for the given problem. The next section provides a mathematical formulation to VRP.

1.2 Problem Formulation

The CVRP is to satisfy the demand of a set of customers using a fleet of vehicles with minimum cost. Achuthan et al (1996) described the problem as follows:

Let

- C= {1, 2,…, n}:the set of customer location.
- 0 : depot location.
- G=(N,E) : the graph representing the vehicle routing network with N={0,1,…,n} and E={(i,j):i,j$\in$N, i<j}.
- q_j : demand of customer j.
- Q : common vehicle capacity.
- m : number of delivery vehicles.
- c_{ij} : distance or associated cost between locations i and j.
- L : maximum distance a vehicle can travel.
- P_j: a lower bound on the cost of traveling from the depot to customer j.
- ℓ(S): lower bound on the number of vehicles required to visit all locations of S in an optimal solution. Note that $S \subseteq C$ and $\ell(S) \geq 1$.
- $\bar{S}$: the complement of S in C
- x_{ij}: 1,2, or 0

The problem is to:

$$\text{minimize } Z=\sum_{i\in N}\sum_{i<j} c_{ij}x_{ij} \quad i\in N,\ i<j \tag{1.2.1}$$

subject to

$$\sum_{i \in C} x_{0i} = 2m \ , \ i \in C \tag{1.2.2}$$

$$\sum_{j<i} x_{ij} + \sum_{i<j} x_{ji} = 2 \ , \ i \in C \tag{1.2.3}$$

$$\sum x_{ij} \leq |S| - \ell(S), \quad i,j \in S, \quad S \subseteq C, 3 \leq |S| \leq n\text{-}2 \tag{1.2.4}$$

$$x_{ij} = 1,2,\text{or } 0 \tag{1.2.5}$$

Constraints (1.2.2) and (1.2.3) known as degree constraints. Constraint (1.2.2) specifies that the number of vehicles leaving and returning to the depot are m. Constraint (1.2.3) specifies that each customer is visited by only one vehicle. Constraint (1.2.4) is referred to as subtour elimination constraints, which prevent subtours from forming loops disconnected from the depot, or eliminate tours that connected to the depot but violate the capacity restriction. Note that a connected component of a weighted or un-weighted graph defined over the set of customers is called a subtour. The subtour will be called a tour if it's connected to the depot in a graph defined over all locations. Constraint (1.2.5) specifies that if a vehicle travel on single trip between i and j then the value of x_{ij} will be 1,and if i=0 and (0,j,0) is a route then the value of x_{ij} will be 2, otherwise the value of x_{ij} will be 0.

1.3 Review and Summary of Thesis

The major focus of this thesis is to develop a hybrid approach to solve CVRPs. We develop and implement methods that combine domain reduction with heuristic algorithms as well as Branch and Cut method.

In this thesis combining domain reduction with a greedy search heuristic (that have restrictions on each route) improves the solution by average of 24% and combining the domain reduction with the Clarke and Wright algorithm improves the solution by average of 8%. When domain reduction combines with branch and cut method, the average time taken to solve the problems have been improved by 49.8%. The thesis illustrates clearly the benefits of using domain reduction to

- Minimizing the cost when combined with a greedy search heuristic algorithm.
- Minimizing the time taken to solve CVRPs when combined with a branch and cut exact method.

The CVRP is a combination of the traveling salesman problem TSP and the bin packing problem **BPP**. The early work of Dantzig et al (1954) on the TSP inspired researchers to develop methods, theories, and constraint to solve the CVRP. In addition, the CVRP formula in Section 1.2 builds on the paper of Dantzig and Ramser (1959b) and used by Laporte et al (1985). Moreover, Fisher (1994a) showed how constraint (1.2.4) can be tightened, while Cornuéjols and Harche (1993) presented two constraints which, have successfully been used to solve CVRP. These constraints are:

Let $W_0, W_1, \ldots, W_i \subseteq C$ satisfy:

- $|W_i \setminus W_0| \geq 1, i=1\ldots,s,$
- $|W_i \cap W_0| \geq 1, i=1\ldots,s,$
- $|W_i \cap W_j| = 0, 1 \leq i < j \leq s,$
- $s \geq 3$ and odd.

The comb inequality is given by:

$$\sum_{p=0}^{s} \sum_{i,j \in W_p} x_{ij} \leq \sum_{p=0}^{s} s|W_p| - \frac{3s+1}{2} + \alpha(\text{m-1}) \qquad (1.3.1)$$

where $$\alpha = \begin{cases} 0, & if\, 0 \notin \bigcup_{i=0}^{s} W_i, \\ 1, & if\, 0 \in W_0 \setminus \bigcup_{i=1}^{s} W_i \text{or } 0 \in W_j \setminus W_0 \text{ for some } j=1,\ldots,s \\ 2, & if\, 0 \in W_j \cap W_0 \text{ for some } j=1,\ldots,s. \end{cases}$$

For the case $0 \notin W_1 \setminus W_0$ the constraints are tightened to

$$\sum_{p=0}^{s} \sum_{i,j \in W_p} x_{ij} \le \sum_{p=0}^{s} |W_p| - \frac{3s+1}{2} + m\text{-}\ell(C \setminus W_1) \qquad (1.3.2)$$

Fisher (1994a) connectivity constraint (1.3.2) tightening can be presented as follows:

$$\sum_{i \in S} x_{0i} + \sum_{i \in S} \sum_{j \in \bar{S}} e_j x_{ij} \ge 2\ell(S) \qquad S \subseteq C \text{ with } |S| \ge 2, \qquad (1.3.3)$$

where

$$e_{j=} \begin{cases} 0, & j \in S, \\ 0, & j \in S' \text{ and } |S'| \le 2, \\ \dfrac{\ell(S)}{\ell(S)+1}, & j \in S' \text{ and } |S'| > 2, \\ 1, & j \in \bar{S} - S'. \end{cases}$$

Constraint (1.3.3) is useful for detecting violating subtour elimination constraints. An alternative version of this constraint was developed by Achuthan et al (1996).

$$\sum_{i,j \in s} x_{ij} + \sum_{i \in s} x_{0i} - m \le |S| - \ell(\bar{S}), \qquad S \subseteq C, 1 \le |\bar{S}| \le n-1. \qquad (1.3.4)$$

We expect that VRP will receive great attention in the coming years due to the following reasons:

- The improvements of TSP techniques.
- The improvements of CP approaches and the increased attentions to combine CP with VRP methods.
- The increased developments in VRP theoretical results.

We review some of the heuristics and the exact methods used to solve the capacitated vehicle routing problem in Chapter 2. Our discussion on heuristics surveys both classical and metaheuristics methods. For the classical methods, we discuss the Clarke and Wright algorithm and the sweep algorithm. Genetic algorithms and simulating annealing are the metaheuristics that are reviewed. Our discussion on exact methods focuses on Branch and Cut.

Chapter 2 also describes the techniques developed over the years to solve constraint satisfaction problems. A comparison between constraint programming CP and operational research OR techniques is provided in this Chapter. The advantages and disadvantages of using either CP or LP to solve optimization problems are discussed.

Chapter 3 develops a simple classical heuristic algorithm for the CVRP. The algorithm is implemented in C++ and applied to solve 10 benchmark CVRP instances. The number of customers for the test problems ranges from 7 to 48. The optimal solutions (that we compared our results to) are obtained using CPLEX. Also the Algorithm results are compared to the results obtained by the Symphony heuristics and the Clarke and Wright (1964) saving Algorithm. Chapter 3 also provides some observations related to domain reduction.

Chapter 4 develops the domain reduction approach to improve the greedy search heuristic algorithm introduced in Chapter 3. Chapter 4 combines domain reduction with the greedy algorithm, the Clarke and Wright algorithm and with a Simulating Annealing metaheuristic algorithm. This Chapter provides conclusions on the effect of domain reduction when combined with different heuristic algorithms. Chapter 5 incorporates Branch and Cut method with domain reduction. The hybrid approach is applied to solve the 10 CVRP literature instances that we used in Chapter 4. A comparison of the results, time taken and gap reduction will follow. Chapter 6 concludes the thesis and provides some suggestions for future work.

An appendix of the literature and generated instances is provided followed by the bibliography.

Chapter 2

Literature review

This Chapter reviews some of the heuristics and the exact methods used to solve the capacitated vehicle routing problem. It surveys both classical and metaheuristics methods. For the classical methods, we review the Clarke and Wright algorithm and the sweep algorithm. For metaheuristics we discuss genetic algorithms and simulating annealing. For exact methods, our focus will be on the Branch and Cut technique. The Chapter shows the developments of Constraint Programming (CP) over the recent years. Also, we review the domain reduction technique. A comparison between constraint programming (CP) and operational research (OR) techniques, is provided with a discution on the advantages and disadvantages of using either (CP) or (LP) to solve optimization problems.

The importance of CVRP in minimizing the cost of the distribution network has motivated many researches in the recent years. Many books, papers and workshops have presented new approaches to solve the VRP and offer a better understanding to the problem. Books like Toth and Vigo (2002), Rayward-Smith et al (1996), Goldberg (1989), Nemhauser (1988) and Golden and Assad (1988) presented the VRP and various techniques to solve it. Further, survey papers like Attanasio et al (2003), Erera and Daganzo (2003), Kleywegt et al (2002), Rousseau et al (1999), Vianna et al (1999) and Prosser and Shaw (1996), offer promising approaches to solve VRPs.

In their paper Garvin et al (1957), discuss the vehicle routing problem in relation to the distribution of gasoline to service stations, using vehicles with different capacities. However, Dantzig and Ramser (1959) developed the first mathematical programming formulation and proposed a heuristic algorithm to solve the vehicle routing problem. Five years later Clarke and Wright (1964) proposed a greedy heuristic that improves the Dantzig and Ramser algorithm. For more details on the methods and techniques to deal with the VRP we refer to the works of Balinski, and Quandt (1964), Bodin, and Golden (1981), Bodin et al (1983), Brodie and Waters (1998), Campos et al (1991), Carpaneto, et al(1989), Christofides (1985), Christofides et al (1981b), Christofides et al (1979), Desrochers et al (1990), Fischetti et al (1994), Forbes et al (1994), Foster, and Ryan (1976), Gaskell (1967), Golden and Assad (1986), Hadjiconstantinou et al (1995), Hall et al (1994), Kolen et al (1987), Kulkarni and Bhave (1985), Lenstra and Rinnooy Kan (1981), Li et al (1991), Magnanti (1981), Naddef (1994), Nelson et al (1985), Paessens (1988), Ribeiro and Soumis (1994), Waters (1988),

The VRP variants mentioned in Table 2.1 are the most basic ones. However, there are many other VRP variants that are more complicated. We refer to the work of Ferland and Mehelon (1988), Gendreau et al (1999), Taillard (1993a) for more details about heterogeneous fleet VRP, Li et al (2007) for details about VRP with multiple vehicle types and Salhi and Rand (1993) for more details about the Vehicle fleet composition problem.

VRP variant	Description	References
Capacitated vehicle VRP	Fleet of vehicles with uniform capacity serves a set of customers with known demands from a single depot.	Augerat et al (1995), Li et al (2005).
VRP with time window	Additional constraint that each customer must be served within a pre-specified time period.	Solomon (1987), Desrochers et al (1992), Zbigniew and Piotr (2002).
Multiple depot VRP	Fleet of vehicles with uniform capacity serves a set of customers from multiple depots.	Salhi and Nagy (1999), Giosa et al (2002).
Periodic VRP	Scheduling is for a fixed number of periods.	Chao et al (1995),Cordeau et al (1997).
Split delivery VRP	The same customer may be served by a number of vehicles.	Archetti et al (2006a), Archetti et al (2006b).
Stochastic VRP	Values for customers and/or demands and/or times are random.	Stewart and Golden (1983), Laporte et al (2002), Bent and Van Hentenryck (2004).
VRP with backhauls	Additional constraint that customers can demand more commodities.	Goetschalckx et al (1989), Kim et al (1997).
VRP with pickup and delivering	Here commodities may be picked up from a certain customer and delivered to other delivery location.	Min (1989), Hernandez and Gonzales (2004), Tang and Galvao (2006)

Table 2.1: Vehicle Routing Problem Variants

In this Chapter we consider the capacitated vehicle routing problem. For convenience we recall the notation introduced in the previous chapter.

- C= {1, 2,…, n}:the set of customer location.

- 0 : depot location.
- $G=(N,E)$: the graph representing the vehicle routing network with $N=\{0,1,\ldots,n\}$ and $E=\{(i,j):i,j\in N,\ i<j\}$.
- q_j : demand of customer j.
- Q : common vehicle capacity.
- m : number of delivery vehicles.
- c_{ij} : distance between locations i and j.
- L : maximum distance a vehicle can travel.
- P_j : a lower bound on the cost of traveling from the depot to customer j.
- $\ell(S)$: lower bound on the number of vehicles required to visit all locations of S in an optimal solution. Note that $S\subseteq C$ and $\ell(S)\geq 1$.
- $\bar{S}$: the complement of S in C
- x_{ij}: 1,2, or 0

The problem as detailed in the previous chapter is to:

$$\text{minimize } Z=\sum_{i\in N}\sum_{i<j}c_{ij}x_{ij} \tag{2.1}$$

subject to

$$\sum_{i\in C}x_{0i}=2m\ ,\ i\in C \tag{2.2}$$

$$\sum_{j<i}x_{ij}+\sum_{i<j}x_{ji}\quad ,\ i\in C \tag{2.3}$$

$$\sum x_{ij}\leq|S|-\ell(S),\quad i,j\in S,\quad S\subseteq C, 3\leq|S|\leq n\text{-}2 \tag{2.4}$$

$$x_{ij}=1,2,\text{or } 0 \tag{2.5}$$

Over the past 40 years, many approaches and solution techniques have been developed to solve VRPs. Some of these approaches are exact like the direct tree search method (Christofides and Eilon 1969), the minimum K-degree centre tree relaxation (Christofides et al 1981a), the set partitioning based method (Agarwal et al 1989), the minimum k-tree relaxation (Fisher 1994 a). Some techniques to solve VRP are heuristics like the Clarke and Wright algorithm (1964), the multi-route improvement algorithm (Thompson and Psaraftis 1993 and Van Breedam 1994), the Fisher and Jaikumar algorithm (1981), the deterministic annealing (Dueck and Scheurer 1990 and Dueck 1993), the Tabu search (Badeau et al 1997, Amberg et al 2000 and Cordeau, Laporte and Mercier 2001), and the Ant system method (Tian et al 2003 and Reimann et al 2004). We will describe some of the heuristics and the exact methods in the following sections.

2.1 Classical Heuristics

Heuristic algorithms to solve VRP have proved to be very useful for solving large problems in reasonable time (Atkinson (1994). Also, heuristics provide good upper bounds that play an important role in exact methods such as branch and cut. Over the last 50 years, many heuristic algorithms had been developed to solve VRP. Classical algorithms and metaheuristics are the classes or the families that the developed algorithms belong to.

Constructive methods were the first category of the classical methods. Building a feasible solution and improving the cost is the idea behind the constructive methods. An example of the constructive method is the Clarke and Wright savings algorithm (1964). The second category of classical heuristics is the two-phase heuristics. In this category, customers are organized into feasible clusters, then the routes constructed for each of them. An example of the two-phase algorithm is the sweep algorithm of Laporte (1992). The following is a brief description for the above mentioned classical algorithms.

2.1.1 The Clarke and Wright Algorithm (1964)

This algorithm is the most popular heuristic for the VRP. The algorithm calculates all the savings s_{ij} between customers i and j. Assuming that c_{i0} is the cost of traveling from the depot to customer i and c_{ij} is the cost of traveling from customer i to j. The following is a description of the Clarke and Wright algorithm to solve the CVRP:

Step 1: Compute the savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ for i,j=1,…,n and $i \neq j$. Rank the savings s_{ij} and list them in descending order**.**

Step 2: Creates the "savings list." Process the savings list beginning with the topmost entry in the list (the largest s_{ij}). For the savings s_{ij} under consideration, include link (i, j) in a route if no route constraints will be violated through the inclusion of (i, j) in a route. The following three cases need to be considered.

Case 1: If neither i nor j have already been assigned to a route, then a new route is initiated including both i and j.

Case 2: If exactly one of the two points (i or j) has already been included in an existing route and that point is not interior to that route (a point is interior to a route if it is not adjacent to the depot in the order of traversal of points), then the link (i, j) is added to that same route. If the point is interior and not violating the capacity then add (i,j) to the same route. If it's violating the capacity make a new route with the point (customer) i.

Case 3: If both i and j have already been included in two different existing routes and neither point is interior to its route, then the two routes are merged by connecting i and j. If they are interior then the merge cannot be done

Step3: If the savings list s_{ij} has not been exhausted, return to Step 2, processing the next entry in the list; otherwise, stop.

Example 2.1

We illustrate the above algorithm using the following CVRP instance:

j \ i	0	1	2	3	4
0		2	3	1	8
1			2	3	4
2				2	6
3					8
4					

Table 2.1.1 Cost Matrix for Clarke and Wright Example

Note that the matrix in table 2.1.1 is symmetric because we are dealing with symmetric CVRP.

The demand is (0,6,10,7,4) units and the capacity is 20 units

Solution: Initial set of routes is Φ

Step 1: Compute the savings

The savings	
1 to 2	2+3-2=3
1 to 3	2+1-3=0
1 to 4	2+8-4=6
2 to 3	3+1-2=2
2 to 4	3+8-6=5
3 to 4	1+8-8=1

Step 2: Creates the savings list

The savings list	
Arc	Associated saving
1 to 4	6
2 to 4	5
1 to 2	3
2 to 3	2
3 to 4	1
1 to 3	0

Step 3:

The first route will be 0-1-4-2-0 and the second route will be 0-3-0. The total cost is 17.

We refer to the work of Altinkemer and Gavish (1990) for more details.

2.1.2 Sweep Algorithm (Wren and Holliday (1972))

In the sweep algorithm each vertex or customer is represented by its polar coordinates. Mathematically, each vertex i will be represented by (θ_i, r_i) where θ_i is the angle for customer i (consider the clock wise) and r_i is the ray length. Start by assigning θ_i^* =0 to an arbitrary vertex i^*, then calculating the rest of the angles from (0, i^*). All the calculated angles will be ranked in an increasing order of their angles. The following steps describe the sweep algorithm:

Step 1: Choose a vehicle v

Step 2: Start from the vertex with the smallest angle, assign vertices to v so that the capacity of the vehicle is not violated.

Step 3: Repeat until all vertices assigned.

Step 4: Solve each route as a traveling salesman problem (TSP) to find the shortest path then stop.

Applying the sweep algorithm to the case of Example 2.1 we get:

Step 1: Choose a vehicle v

Step 2: Start with 0-3 then 3-2. Note that the total demands of customers 2 and 3 is 17, this means that the route cannot have any more customers.

Step 3: Choosing the next vehicle and repeating Step 2. Route 2 will be 0-1-4-0. The total cost will be 14.

Note that Example 2.1 has 4 customers only. For this reason, Step 4 is not needed.

Wren and Holliday (1972) presented a different way to calculate the polar angle that considers the configuration of the points around each depot (clock wise). The new ordering then used to generate four different initial solutions by assigning customers (in their paper they used cities instead of customers) starting from four different positions in the ordered list. The best of these four solutions is chosen as an input to an improvement phase. This latter phase uses seven procedures repeatedly until no improvement can be done. Accurate results are reported on two problems having two depots and up to 176 customers.

2.2 Metaheuristics

The quality of the solution obtained by any of the metaheuristic algorithms is normally far better than the one obtained by the classical algorithms since metaheuristic algorithms explore deeply all the solution space. However, metaheuristics take more time than the classical heuristics. The following details two popular metaheuristics:

2.2.1 Simulating Annealing (SA)

As a stochastic relaxation technique, SA has its origin in statistical mechanics. The process of crystallizing a solid by heating it to a high temperature and gradually cooling it down motivates the development of SA. The SA algorithm was introduced by Metropolis et al. (1953). Assuming $\Delta = f(x) - f(x_t)$, where $f(x)$ is the best value for the objective function found so far, and $f(x_t)$ is the value of the objective function at iteration t. The solution will be accepted as a new current solution if $\Delta \le 0$. If $\Delta > 0$, any moves with a probability of $e^{-\Delta/T}$ that increase the objective function are accepted, where *T* is the temperature and its value varies from large to close to zero. The values of *T* are controlled by a cooling schedule that specifies the temperature values at each stage. Zbigniew and Piotr (2002) proposed that a solution ***x*** is drawn randomly in $N(x_t)$ at iteration ***t***. If $f(x) \le f(x_t)$, then x_{t+1} is set equal to x; otherwise

$$x_{t+1} = \begin{cases} x \text{ with probability } & p_t \\ x_i \text{ with probability} & 1 - p_t \end{cases}$$

where p_t is a decreasing function of ***t*** and of $f(x) - f(x_t)$.

The SA stops when:

- The value f^* has not decreased by $\pi_1\%$ for at least k_1 consecutive cycles of ***T*** iterations;
- The number of accepted moves has been less than $\pi_2\%$ of ***T*** for k_2 consecutive cycles of ***T*** iterations;
- k_i of ***T*** iterations have been executed.

where π_1, π_2 and k_i are pre-specified values.

The application of SA to solve CVRP is to take an initial solution to the problem and consider it as the best solution. A neighborhood search of removing and adding customers from the routes follows. The adding and removing is a random process within the above mentioned boundaries. Updating the best solution as the cost is reduced.

Zbigniew and Piotr (2002) use a parallel SA approach to solve the Solomon (1987) set of problems. The set of problems is 54 instances each with 100 customers. The obtained results were close to optimal and better than any other algorithm used to solve the same set. SA proves to be an accurate method when used to solve VRP.

2.2.2 Genetic algorithms (GA)

Inspired by the biological evolutionary, Fraser (1957) proposed a computer simulation of evolution. The algorithm represents the solution as a population of chromosomes $X^1=\{X_1^1,...,X_N^1\}$,where *N* is the number of vertices or customers. Then

- Select two parent chromosomes from X^1.
- Use the parent chromosomes to produce offspring that forms the next generation.
- Mutate randomly each offspring with a small probability.

The above three steps will be repeated K times for each iteration t=1,…,T , where $\mathrm{k} \leq N/2$ and T is the number of generations. Then the next step will be applied:

- Create X^{t+1}, from X^t. This will be done by removing the 2k worst solutions in X^t (the ones with the highest cost) and replacing them with the 2k new offsprings.

In order to apply the genetic algorithm to solve CVRP, the following must be considered:

- Good genetic representation. This means the number of vehicles (routes) must be specified.
- Initial population constructor. This means initial solution to the problem must be provided.
- Determine fitness, crossover and mutation operators. This means a criterion for improving the solution must be specified.

Now the genetic algorithm will repeat the following for pre-specified number of iterations:

- Choose two customers.
- Use the two customers to form a route without violating the capacity.
- Repeat until all customer demands are satisfied.
- Use the fitness, crossover and mutation operators to improve the solution.

Berger and Barkaoui (2004) proposed a parallel hybrid algorithm to solve 56 benchmark problems of Solomon (1987). Each problem involves 100 customers, randomly distributed over a geographical area. The computational results showed that the algorithm is cost-effective and very competitive to the best known solution, and generated six new best-known solutions for the Solomon sets.

2.3 Branch and Bound

Branch and Bound (BB) is a systematic method for solving optimization problems. Presented by Land and Doig (1960), BB was developed to solve general discrete programming problems and mixed discrete programming problems. Assuming that the problem is a minimization problem the branch and bound procedure minimizes a function of the variables over a region of feasible solutions. The main components of branch and bound can be described as follows:

- An upper bound that is obtained by the application of a heuristic. It is important to start with a tight upper bound on the problem.
- Problem relaxation. Relaxing the original problem by excluding some constraints. Problem relaxation normally provides a tight lower bound.
- The branching rule. This represents the way to separate the sets.

Let S be the set of feasible solution and T be a superset of S. T is obtained by excluding one or more constraints from S. The following branch and bound algorithm steps are as described by Balas and Toth (1985):

Step 1: Set S_0=T the superset of S and U=∞ as the upper bound. Create a list of active nodes where entries in the list consist of a lower bound L_i and a set S_i. Initialize the list with initial lower bound L_0 and initial set S_0.

Step 2: Stop if there are no entries in the list. If U=∞ then there is no feasible solution to the original problem, else the stored solution is the optimal solution and U is the optimal value. Otherwise, if there are entries in the list choose the entry from the list, say S_i and solve the subproblem.

Step 3: If $L_i \geq U$, then discard S_i and go to Step 2.

Step 4: If the solution to the subproblem is also a solution to the original problem then set U= L_i and store the solution. Go to Step 2.

Step 5: Separate the feasible set of solutions S_i into smaller subsets $\{S_{i1}', S_{i2}', \ldots, S_{in}'\}$ by the prescribed branching rule where

$$\bigcup_{i=1}^{n} S_{ij}' = S_i .$$

Step 6: Set the lower bounds L_{ij}' on the objective function value over each set S_{ij}' to be equal to L_i. Go to step 2.

The following example illustrates the algorithm:

Example 2.2: Consider the minimization problem

$$\text{Min } 8x_1 + 11x_2 + 6x_3 + 4x_4 \tag{2.2.1}$$

Subject to

$$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \tag{2.2.2}$$

$$x_j \in \{0,1\} \quad j=1,2,3,4 \tag{2.2.3}$$

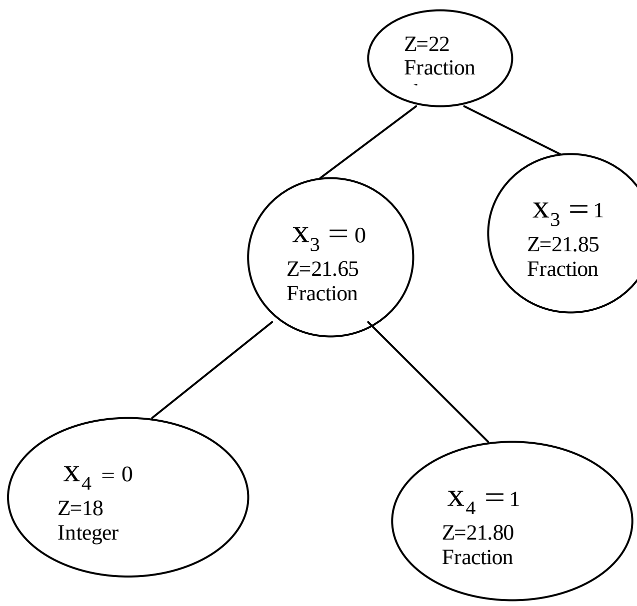
Solving the LP relaxed problem where (2.2.3) replaced by $x_j \le 1$ for all j, yields the solution: x_1=1, x_2=1, $x_3 = \frac{1}{2}$, x_4=0. The objective function value is 22. It's clear that the LP solution is not satisfying constraint (2.2.3), since $x_3 = \frac{1}{2}$ is not integer. In order to force x_3 to be integer, the branching process is applied on x_3 this creates two new problems, one with x_3=0 and the other with x_3=1. Solving the relaxed sub-problems we get:

x_3=0: x_1=1, x_2=1, x_4=0.667, with objective value 21.65

x_3=1: x_1=1, x_2=0.714, x_4=0, with objective value 21.85.

Since the problem is a minimization problem the solution with the lowest objective value should be chosen. So we take the sub-problem with x_3=0. Observing that the value of x_4 is not an integer, the branching process is applied again. This results two sub-problems, one with x_4=0 and one with x_4=1. The procedure continues until all constraints are satisfied and all the values of x_j, j=1,2,3,4 are integers. Figure 2.3.1 illustrates the search tree.

Figure 2.3.1: Branch and Cut Search Tree



Branch and Bound is one of the good methods to find the optimal solution (Malik, and Yu (1993)). However, the method can take a long time and could lead to exponential time complexities in the worst cases (Khoury and Pardalos (1995)).

The next Section provides the cutting plane technique. This technique minimizes the domain and sometimes accelerates the search.

2.4 Cutting Plane Technique (Cornuejol 2007)

Ralph Gomory introduced the cutting plane method to solve ILP and to solve general **convex optimization** problems (Boyd (1994)). The method consists of polyhedral cutting planes. The idea behind the cutting plane technique is to generate cuts until a best or an optimal solution is obtained. Figure 2.1 illustrates the method.

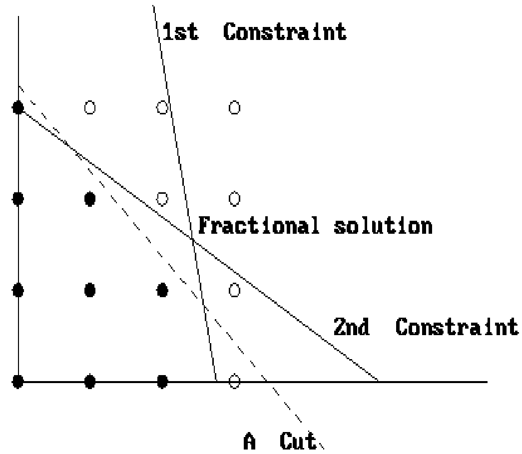


Figure 2.1 Gomory cut (A Gomory cut (1998))

The method can be described as follows:

- Solve the LP relaxation of the problem.
- If the result is integer then it will be the optimal solution and no further work is required.
- If the result of solving the LP relaxation is non-integer, then using the LP relaxation solution Gomory cuts are generated as we will show in the next example.
- Add the generated cut to the problem as a constraint then repeat the procedure starting from the first step.

The following example illustrates the cutting plane method:

Example2.3: Consider the following integer minimization problem

$$\text{Min } 7x_1 + 9x_2 \tag{2.3.1}$$

Subject to

$$-x_1 + 3x_2 \le 6 \tag{2.3.2}$$

$$7x_1 + x_2 \le 35 \tag{2.3.3}$$

$$x_1,\ x_2 \text{ positive integers} \tag{2.3.4}$$

Solving the relaxed problem yields:

Variable	x_1	x_2	s_1	s_2	-Z	RHS
x_1	0	1	$\frac{7}{22}$	$\frac{1}{22}$	0	$\frac{7}{2}$
x_2	1	0	$\frac{-1}{22}$	$\frac{3}{22}$	0	$\frac{9}{2}$
-Z	0	0	$\frac{28}{11}$	$\frac{15}{11}$	1	63

Table 2.2 optimal tableau

From Table 2.2 the first constraint will be:

$$x_2 + \frac{7}{22}s_1 + \frac{1}{22}s_2 = \frac{7}{2} \tag{2.3.5}$$

Putting all the integer parts in one side and the fractional in the other side we get:

$$x_2 \text{-}3 = \frac{1}{2} - \frac{7}{22}s_1 - \frac{1}{22}s_2 \tag{2.3.6}$$

It’s clear that the right hand side must be integer since the left hand side is integer. Also, since $x_2 \le 1$ then the right hand side is negative as the left hand side is negative. Hence we can get the following constraint:

$$\frac{1}{2} - \frac{7}{22}s_1 - \frac{1}{22}s_2 \le 0 \tag{2.3.7}$$

In the current solution $s_1 \text{and} s_2$ are zero, which means that (2.3.7) is violated. Constraint (2.3.7) is a cut and it can be added to the original problem. The process will continue until we have an integer solution.

The method when applied to some ILP or MILP problems may generate cuts in a way that the newly generated cut will result in little improvement from the previous cut. Hence the majority of the earlier researchers avoided using the method until Padberg and Rinaldi (1987) highlighted the benefit of combining the method with Branch and Bound to solve the TSP. The Branch and Cut method used the strength of Cutting Plane techniques to cover the weakness in Branch and Bound.

2.5 Application of Branch and Cut Method to VRP

The term firstly coined by Padberg and Rinaldi (1987) in their paper on the TSP. The term Branch and Cut refers to Branch and Bound (BB) and Cutting plane techniques. The following are some well-known approaches of branch and cut method to solve the VRPs.

2.5.1 The Laporte et al (1985)

Laporte et al (1985) used a Branch and Cut method to solve CVRP subject to distance and capacity restrictions. For Euclidean problems, they considered VRP with symmetric graph G=(N,E),where N is a set of nodes that may represent customers or cities and E is a set of undirected edges. The distance matrix associated with the edges is C (c_{ij} or c_{ji}) whenever i>j. C satisfies the triangle inequality $c_{ij} \leq c_{ik} + c_{kj} \quad (i,j,k \in N)$. Laporte et al (1985) also assumed that all vehicles have the same capacity. This formulation was:

Formulation:

$$\text{minimize } Z=\sum\sum c_{ij}x_{ij} \quad i \in N,\ i<j \tag{2.5.1}$$

subject to

$$\sum_{i \in C} x_{0i} = 2m \ ,\ i \in C \tag{2.5.2}$$

$$\sum_{j<i} x_{ij} + \sum_{i<j} x_{ji} \quad ,\ i \in C \tag{2.5.3}$$

$$\sum x_{ij} \le |S| - \ell(S), \quad i,j \in S, \quad S \subseteq C, 3 \le |S| \le n\text{-}2 \qquad (2.5.4)$$

$$x_{ij} = 1,2,\text{or } 0 \qquad (2.5.5)$$

where constraints (2.5.2) and (2.5.3) known as degree constraints. Constraint (2.5.2) specifies that the number of vehicles leaving and returning to the depot are m. Constraint (2.5.3) specifies that each customer is visited by only one vehicle. Constraint (2.5.4) is referred to as subtour elimination constraints, which prevent subtours from forming loops disconnected from the depot, or eliminate tours that connected to the depot but violate the capacity restriction. Note that a connected component of a weighted or un-weighted graph defined over the set of customers is called a subtour. The subtour will be called a tour if it's connected to the depot in a graph defined over all locations. Constraint (2.5.5) specifies that if a vehicle travel on single trip between i and j then the value of x_{ij} will be 1,and if i=0 and (0,j,0) is a route then the value of x_{ij} will be 2, otherwise the value of x_{ij} will be 0.

Algorithm:

The algorithm to solve the above Euclidean VRP developed by Laporte, Nobert and Desrochers (1985) can be described in the following 10 steps:

Step 1-Solve the problem using simplex method to obtain $\overline{Z}$,where $\overline{Z}$ is the solution for the relaxed problem.

Step 2-Compare $\overline{Z}$ with the cost of best solution Z*. If $\overline{Z} \ge Z^*$ update the list of sub-problems and choose the next sub-problem then start from step 1.Otherwise continue.

Step 3-Force the variables that are not in the subtour to zero using subtour prevention constraints.

Step 4-Purge ineffective constraints.

Step 5-Generate distance and capacity constraints.

Step 6-Generate Gomory cuts.

Step 7-Apply Branching procedure. If the solution is integer then update Z* and continue. Otherwise continue.

Step 8-Backup search tree.

Step 9-Update the list of problems.

Step 10-End the algorithm if the list of sub-problems empty. Otherwise choose the next sub-problem and repeat the procedure.

When the problems are non-Euclidean, Laporte et al (1985) modified the algorithm and the formulation for the Euclidean problems. Forcing certain rules on the edge x_{ij}, i<j to be defined in the formulation. Also, replacing the subtour elimination constraint by $\sum_{i \in S} x_{0i} + 3 \sum_{E(S,\bar{S})} x_{ij} \geq 4$, $3 \leq |S| \leq$ n-2.

Laporte et al (1985) used Branch and Cut method to solve CVRP both Euclidean and non-Euclidean. Their test problems ranged from 15 to 50 customers for the Euclidean type and from 15 to 60 customers for the non-Euclidean assuming that the number of used vehicles is free. For each problem size they generated three problems. To determine the problems characteristics, the three generated problems were tested using different combinations of maximum vehicle capacity and maximum traveling distance for each vehicle.

Laporte et al (1985) tested their algorithm on a CYBER173 computer, using Fortran FTN5 compiler. They used the Land and Powell (1973) LP solution routine. They allowed each problem a running time of 500 seconds. Laporte et al (1985) showed that solving non-Euclidean problems is much easier than solving the Euclidean ones and the obtained results were far better than those obtained by using branch and cut and cutting plane separately in terms of accuracy.

Figure 2.1 and Figure 2.2 are the flow charts of the Laporte et al. (1985) algorithm for Euclidean and non-Euclidean problem:

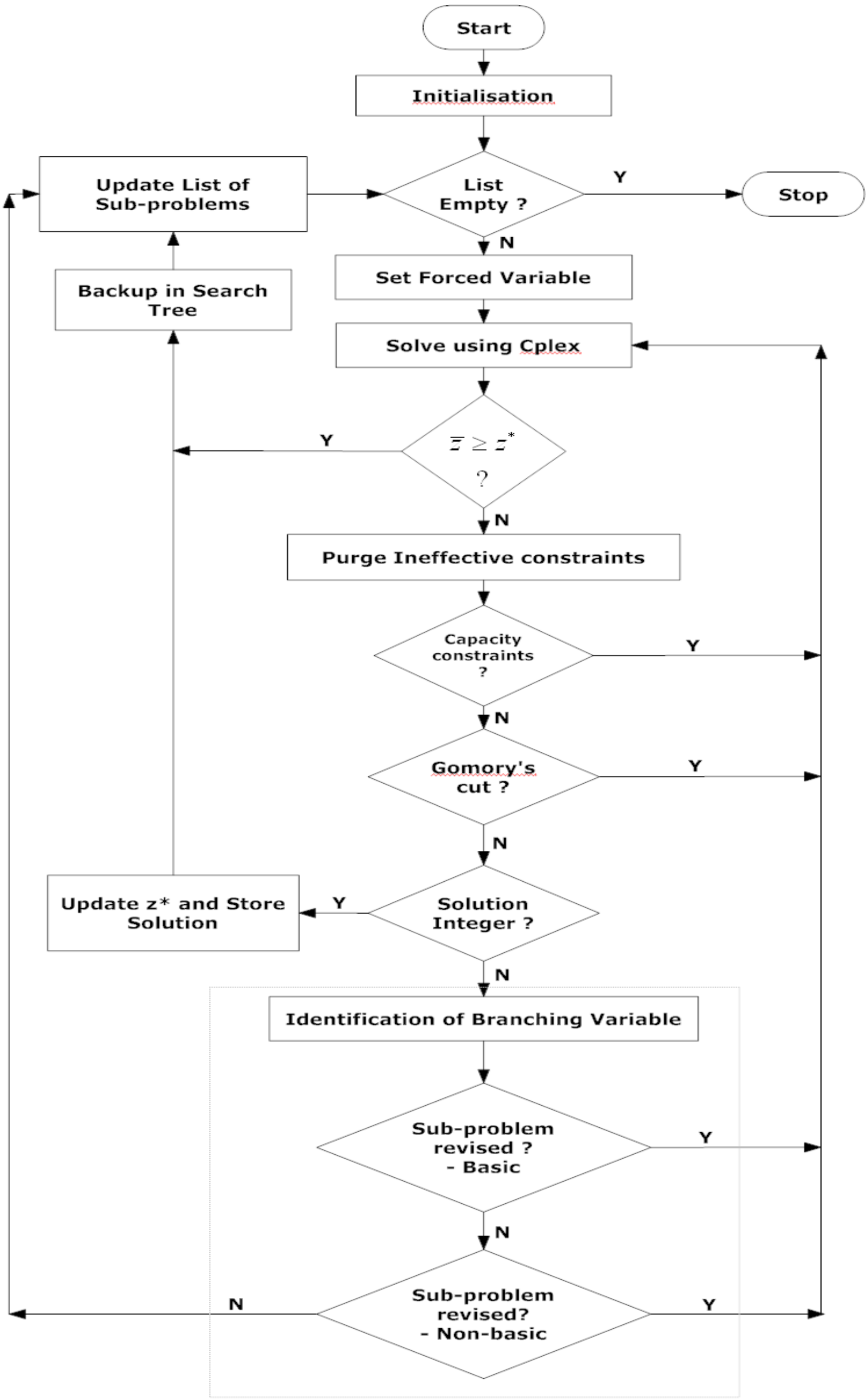


Figure 2.1: Algorithm for Euclidian CVRP

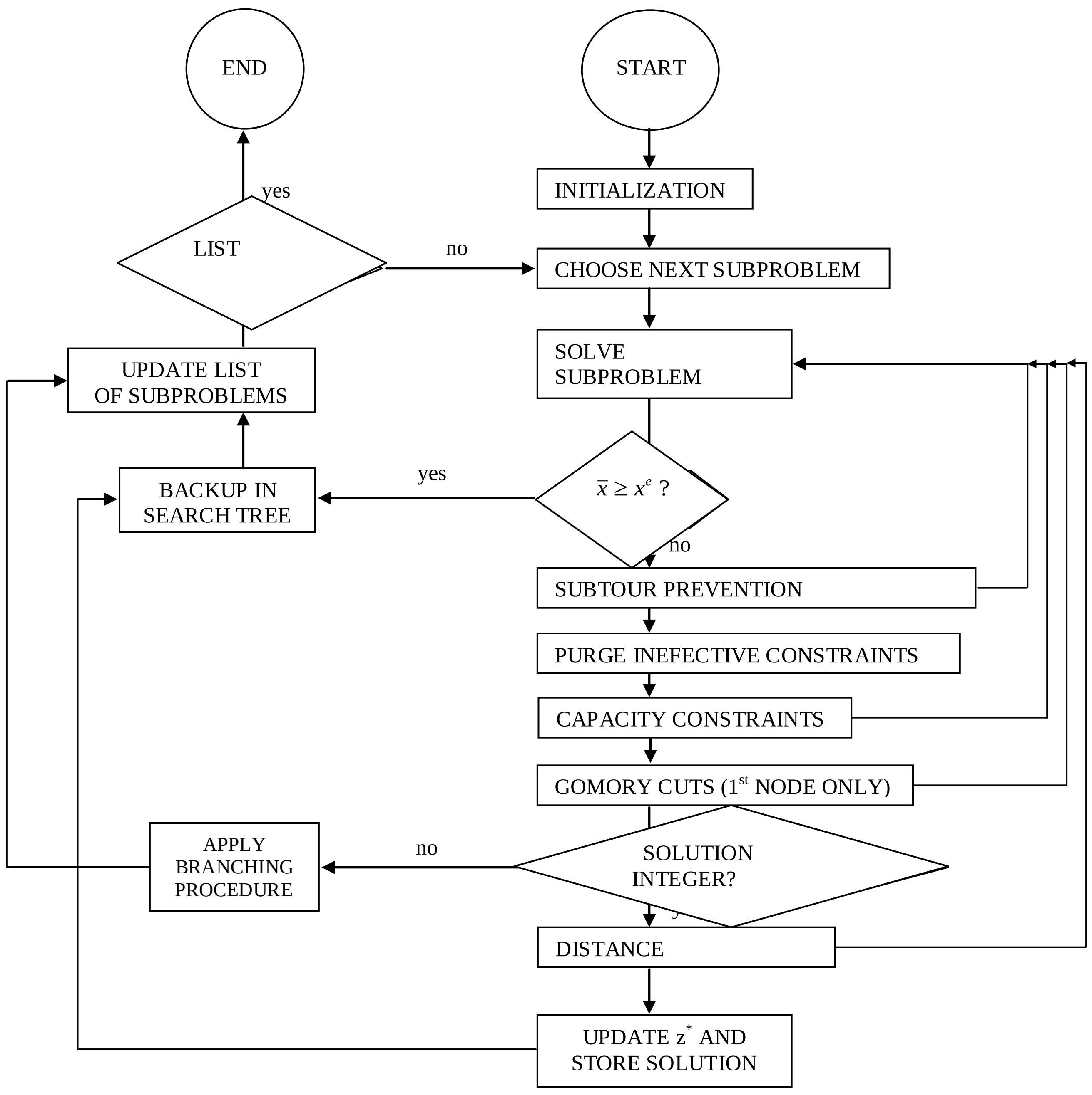


Figure 2.2: Algorithm for Non-Euclidean Problems.

2.5.2 Achuthan et al (2003) Improved Branch and Cut Algorithm.

Achuthan et al (2003) proposed several new cutting planes for capacitated vehicle routing problem. The proposed cutting planes used in the branch and cut algorithm were tested on 1,650 simulated Euclidean problems as well as 24 standard literature problems. The problems ranged from 15-100 customers. The results obtained by the improved branch and cut algorithm were more accurate with reasonable time taken to solve the problems.

Achuthan et al (2003) also, developed a number of search procedures to identify violations to the problem constraints. The following is a brief summary of their work.

Consider the CVRP formulation mentioned earlier in this Chapter. Achuthan et al (2003) presented new cuts described in the following results:

Theorem 1: Let $S, T_1, T_2, \ldots T_k \subseteq C$ be such that

a) $k \geq 2$ and $\sum_{i \in S \cup T_p \cup T_q} q_i > Q$ for every $1 \leq p \neq q \leq k$;

b) $T_i \cap T_j = \phi$ for $i \neq j$;

c) $S \cap T_i = \phi, 1 \leq i \leq k$;

d) $T = \bigcup_{i=1}^{n} T_i$.

Then, for any feasible solution (x_{ij}) of the CVRP we have

$$3\sum_{i,j \in S} x_{ij} + \sum_{E(S,T)} x_{ij} + \sum_{p=1} \sum_{i,j \in T_p} x_{ij} \leq 3|S| - 2 + |T| - k \, . \tag{2.5.7}$$

Corollary 2: $T_1, T_2, T_3 \subseteq C$ satisfy the hypothesis of theorem 1. Then, for any feasible solution (x_{ij}) of the CVRP we have

$$2\sum_{i,j \in S} x_{ij} + \sum_{E(S,T)} x_{ij} + \sum_{p=1}^{3} \sum_{i,j \in T_p} x_{ij} \leq 2|S| + |T| - 4 \tag{2.5.8}$$

Theorem 3: There exists an optimal solution X= (x_{ij}) of the CVRP satisfying the following constraints:

$$\sum_{i,j\in S} x_{ij} + \sum_{j\in S} x_{1j} \le |S| + 1 \quad \text{for all } S \subseteq C \text{ and } \sum_{i\in S} q_i \le Q, \tag{2.5.9}$$

Q is vehicle capacity

Theorem 4: There exists an optimal solution X= (x_{ij}) of the CVRP satisfying (2.5.9) and the following constraints:

$$\sum_{i,j\in S} x_{ij} + \sum_{j\in S} x_{1j} \le |S| + \left\lfloor \frac{2(\sum_{i\in S} q_i + \delta)}{Q+1+\delta} \right\rfloor \quad , \text{ for all } S \subseteq C \text{ with } 2 \le |S| \le |C| \text{ and}$$

$$\sum_{i\in S} q_i > Q \tag{2.5.10}$$

$\delta = 0,1$ according as Q is odd or even

Corollary 5: There exists an optimal solution X= (x_{ij}) of the CVRP with variable m satisfying

$$m \le \begin{cases} 1, \text{ if } \sum_{i\in C} Q \\ \min\left\{ n, \frac{2(\sum_{i\in C} q_i + \delta)}{Q+1+\delta} \right\}, \text{othrwise} \end{cases} \tag{2.5.11}$$

Where $\delta = 0,1$ according as Q is odd or even

In their paper, Achuthan et al (2003) used six searching procedures to detect violations. The first search was that introduced by Laporte et al (1985), the second and the third searches were a modification of Achuthan et al (1996). Others were developed to detect violations either to the elimination constraint used by Laporte et al (1985) and Achuthan et al (1996) or to the proposed cutting plane.

Achuthan, Caccetta and Hill (2003) applied the algorithm to solve 24 benchmark problems. Three of these problems were Christofides (1969), four of them were Christofides (1979), and the rest were Fisher (1994a) and Reinelt (1981). The algorithm solves three problems optimally when single routes allowed and 4 of the problems had been solved optimally when single routes were not allowed. In general the algorithm provides better results than the known solutions at the time.

As any exact method branch and cut has advantages and disadvantages. The following section explains some of the advantages as well as disadvantages in using branch and cut method to solve the LP problems.

2.6 The Advantages and Disadvantages of Branch and Cut

When Branch and Cut was first used to solve VRPs, it was clear that the method performance was good (Araque (1989), and Araque et al (1994)). The Branch and Cut method improved rapidly in recent years especially when dealing with VRPs. The improvement of the method and the successful use of its applications to solve VRP encouraged researchers to use it in solving large scale Symmetric TSPs in recent years. As any exact method, the Branch and Cut method has strengths and weaknesses, also using it will result advantages and disadvantages. The advantages of using Branch and Cut method can be outlined as follows:

- Using valid cutting planes present in the LP will save enormous time.

- In terms of memory allocation, large savings are made by using the constraints present in the original linear program LP from previous lower bound generations.

- By branching, the method overcame the problem of generating cuts in a way that the newly generated cut might be the same or slightly different than the previous one.

- Generating cuts and adding the violating ones to as a constraint to the original problem will accelerate the search for the optimal solution.

The disadvantages of using the method can be described as follows:

- The method removes constraints from the LP tableau as the process continues searching for the optimal solution. By doing this the method saves time and memory. However, removing the constraints from the LP tableau (in some cases) may be too early and the lower bound may not be too high. Therefore regenerating the early removed constraints may be essential in a certain stages of the process. Laporte, Nobert and Desrochers (1985) and Achuthan, Caccetta and Hill (2003) have shown that constraints rarely need to be regenerated for the CVRP.

- At certain stages of the process and for some problems, exploring a node that has different restrictions to the node which was previously explored can result many non-tight constraint in the LP may and poor initial lower bound.
- As part of the process removing child nodes from the list and then generating lower bound, the generated lower bound may be greater than the lower bound value stored when the child node was placed on the list. This is due to the use of different constraints in the LP.

2.7 Constraint Programming (CP)

Constraint Programming (CP) (also called Constraint Logic Programming) is the embedding of constraints in a logic programming language. The CP method based on the idea of using logic to satisfy a large number of constraints (Hooker (2005)). In the seventies, Artificial Intelligence researchers studied constraint satisfaction problems. However, it was in the eighties that the first systematic use of the constraint programming emerged (Roman Barták(1998)). In the following years CP techniques improved rapidly. As computers become faster and the world advanced in terms of knowledge, CP expanded it applications to solve various real life problems. Natural language processing, operations research, computer graphing and molecular biology are examples of the new domains CP expanded its application to (Hooker (2002)).

The early work of Waltz (1972) and Montanari (1974) on picture processing inspired Artificial Intelligence researchers to develop logical-algorithms to satisfy the constraints of certain problems. Constraint satisfaction problems can be seen in almost all the real life sectors. For example:

- graph coloring
- analysis and synthesis of analog circuits

- option trading analysis
- cutting stock
- DNA sequencing
- scheduling
- chemical hypothetical reasoning
- warehouse location
- forest treatment scheduling
- airport counter allocation
- puzzles like crosswords and N-queen.

Constraint satisfaction problems normally consist of finite variables with finite domains and finite constraints restricting the values of the variables. The problem solution will involve the use of logic to assign the variables with values from the domain so that all constraints are satisfied.

Mathematically in most of the cases, solving constraint satisfaction problems using logic algorithms will result in feasible solutions that are not optimal. The following are some techniques to solve constraint satisfaction problems:

2.7.1 Binarization of Constraints

The constraint satisfaction problem can be presented as a set of nodes. Each arc represents a constraint. If the originating and terminating nodes of an arc are the same, the node is called unary constraint, such constraints can be satisfied by reducing the domain. Thus, any problem with unary constraints can be converted to a binary constrained problem. The general approach to converting a constraint satisfaction problem to binary problem is:

- Minimize the set of constrained variables in the problems by assigning Cartesian product domain. The summarized variables will be called encapsulated variables using a valid domain reduction technique.
- Reduce the encapsulated domain.
- Combine the resulting individual solutions to the solution of the constraint system. This could be achieved by either hidden variable encoding or dual encoding.

2.7.2 Systematic Search Algorithms

Although taking a very long time to process the problem, systematic search algorithms were used more often in solving constraint satisfaction problems due to their ability in finding a solution or at least proving that there is no solution to the given problem. One of the following two approaches must be followed in order to develop a systematic algorithm:

Generate and Test (GT)

Algorithms in the GT approach start firstly by guessing solutions to the given problem, then testing if these solutions satisfy the problem constraints. Note that the method takes the first correct solution that satisfies all the problem constraints also, it rejects the guessed solution with all the values assigned to the variables even if one value violates a certain constraint.

Backtracking (BT)

Backtracking algorithms are the most powerful systematic search method used to solve constraint satisfaction problems. As in the generate and test method (GT), Backtracking starts by guessing solutions then testing one solution after the other. The testing procedure based on checking constraint(s) violations caused by the values assigned to the variables. Unlike GT the method will keep changing the violating values only.

2.7.3 Consistency Techniques

First introduced by Waltz (1972), consistency techniques are efficient in ruling out inconsistent possibilities in the domain. The techniques are normally used combined with other constraint programming or operational research techniques and rarely used alone. The consistency of constraint satisfaction problems may be reached using one of the following techniques.

- **Node Satisfaction Technique**

This technique is easy to understand and simple to use. The variables in this technique are represented by nodes. A node will be called node consistent if every value assigned to the variable satisfies all constraints. In case there is an assigned value that does not satisfy a certain constraint, the assignment will fail and the assigned values will be removed from the domain.

- **Arc Consistency Technique**

This technique treats each constraint as an arc connecting the nodes that normally represent variables. The arc will be called arc consistent if for every value in the domain of the first node there is a value in the second node domain such that both values don't violate any constraint. All the violating values in the first node domain will be removed. Note that if a_i, a_j are two nodes and the arc (a_i, a_j) is consistent, it doesn't mean that arc (a_j, a_i) also consistent.

- **Path Consistency Technique**

The test for consistency using the arc consistency technique on two or more arcs will lead to the removal of a large number of values. Path consistency is a more efficient technique in detecting inconsistency and removing inconsistent values. In this technique any node with arc consistency (all arcs associated with the node are arc consistent) is called restricted path consistent. This means a node a_i will be called restricted path consistent if (a_i, a_j),(a_i, a_k) are arc consistent also if (a_i, a_m) a non consistent arc does not exist. Clearly if (a_i, a_m) exists it will be removed by the method.

2.7.4 Constraint Propagation

Constraint propagation is a technique to solve constraint satisfaction problems by combining systematic search and consistency techniques. To develop a constraint propagation algorithm, one of the following approaches is adopted.

- **Backtracking Search**

The method is a combination of Arc consistency and Backtracking; it starts by guessing solutions then test the guessed solution for Arc consistency.

- **Forward Checking**

This method uses restricted arc consistency between the current variable and the future variables.

- **Look Ahead Search**

Unlike forward checking, this method doesn't look for restricted arc consistency between the current variable and the future variables only but also performs full arc consistency search.

2.7.5 Value and Variable Ordering

This search method requires the specification of the order of variables and the order of the values assigned to each variable.

- **Variable Ordering**

The order of the variables may be static or dynamic i.e. either the order of the variable is found before the search and this ordering is kept until the end or at each point of the search the next variable must be specified.

Value ordering

After determining the order of variables, the order of the values that must be assigned to each variable also may be detrained in this method to solve the constraint satisfaction problems. The most common heuristics to determine the values are based on the principle of succeed first, where choosing the value of each variable tested by the constraints and the first succeeded value taking the first order and so on.

2.7.6 Reducing Search

The idea behind this method is to reduce the domain and eliminate the need for backtracking. The most common techniques to perform the reducing search are cycle-cutset and **MACE**.

- **Cycle –Cutset**

This method maintains variable consistency to cut all the cycles in a graph. This may help finding the ordering of the rest of variables without needing the backtrack procedure. The next step in this method is to extend the partial solution to a complete solution.

- **MACE**

Named after the American computer scientist McCune (2003). This method maintains arc consistency in order to cut all the cycles in a constraint graph.

2.8 Constraint Programming and Operations Research

Constraint Programming (CP) and Operation Research (OR) techniques have provided many solution algorithms to various optimization problems over the years (Hooker 2007). The strengths of CP and OR algorithms can be seen through the solutions and the time taken to perform the search. However CP and OR algorithms have some weakness in processing large scale problems or NP-hard problems. Hooker (2002) showed that most of the CP and OR algorithms weaknesses can be covered by combining the two approaches together. CP algorithms can find a feasible solution to an optimization problem within reasonable time but such solution is rarely optimal. In theory OR algorithms are able to find an optimal solution for most of the optimization problems but the time taken to find it may be very long in most cases. Hence, combining CP algorithms with OR algorithms to solve an optimization problem may find an optimal solution within a reasonable time. Although developed by researchers with different scientific background to solve different kinds of problems, CP and OR sharing almost the same search approaches to solve problems. Table 2.1 provides more details.

CP	**OR**	**Search Method(s)**	**Comments**
Systematic search	Branch and Bound	Branching	Both CP and OR methods relay on branching to search for the solution.
Domain reduction and constraint propagation	Cutting Plane and Benders cuts	Inference	To minimize the solution domain CP uses domain reduction and constraint propagation while OR uses cutting planes and benders cut approach.
Constraints Store	Continuous relaxation	Relaxation	CP keep tracks of feasible solution using constraint store while continuous relaxation is so important to solve problems using OR algorithms.
Constraint Store and Domain Reduction	Continuous Relaxation and Cutting Plane	Strengthen relaxation by inference	CP strengthens the constraint store by reducing variable domains while OR strengthen the continuous relaxation by adding cut.

Table 2.1: A Comparison Between CP and OR

2.9 Integrating CP and OR Techniques

In recent years, many researchers have tried to introduce a unifying scheme to combine CP with OR techniques (Hooker 2007). Using different solving methods and different problems, most of the papers provided good results and most of them chose at least one of the following approaches:

- **Double modeling**

This approach writes the problem as a constraint satisfaction problem. The problem can be solved using CP techniques and also writes the same problem as an optimization problem that can be solved using OR techniques. While solving the problem, the two models will exchange information to accelerate the search for an optimal solution.

- **Search and Infer Duality**

This approach normally examines all possible solutions (CP techniques may be used), if none of the solutions are optimal then it will start branching (OR techniques may used). Then an inference process will start by reasoning facts from the constraints.

- **Decomposition**

Using Bender's decomposition, the problem may decompose into a master and sub-problems each with variable domain. The master problem will perform the search over some of the problem variables, while the sub-problem will solve the given problem using the remaining variables and by the information obtained from the master problem.

- **Relaxation**

This approach uses an OR relaxation technique(s) combined with search and infer or with the decomposition approaches. Relaxing the problem will prune the search tree and accelerate the search and for the decomposition approach it will improve the sub-problem decomposition.

2.10 Constraint Programming and VRP

Commercially there are several software packages to solve VRPs using CP(ILOG Dispatcher 4.0, ILOG Solver 6.0, etc…) These packages according to Kilby, Prosser and Shaw (1998) still require additional features to perform the search, as they don't have the following:

- The ability to geo code the addresses.
- A graphical user interface for displaying routes.
- The ability to calculate distance and time traveled from one map point to another.
- The ability to change routes manually.
- A method of easily specifying and entering constraints.
- Interfacing with other systems.

The pruning achieved through propagation attracted an increase attention to use CP to solve VRPs. On the other hand, OR methods had been proven efficient in solving VRPs (Baldacci and Mingozzi (2006)). Combining CP with OR approaches may seems an excellent approach to deal with VRPs. However, the natures of the search procedures for CP and OR may cause an important problem that must be overcome. The CP basic principle **chronological backtracking** means that all decisions must be undone in the reverse of the order they were made. On the other hand, OR methods may assign a customer to a route then in the process it removes this customer and replaces it by another one. Then because of chronological backtracking to undo this customer and replace it by another one, all operations performed since that time must be undone as well. Kilby et al (1998) proposed two ways to overcome this problem. The first is to use the constraint system as a rule checker by allowing a heuristic or meta-heuristic to control search. The second way is wrapping up local search changes within an operator to insulate the Constraint Programming system from the changes being made at the lower level.

Kilby et al (1998) also suggested that using constraint programming alone to solve VRPs will provide feasible solutions without considering the objective function.

Caseau et al (2001) proposed a hybrid algorithm that combines a genetic algorithm with CP. The hybrid algorithm has been applied to solve Solomon (1987) benchmark problems. The obtained results were close to the best known solutions and the time taken to solve the problems using the hybrid algorithm was less.

2.11 Advantages and Disadvantages of Integrating CP with OR

There are several advantages provided by CP and OR integrated algorithms. The advantages are:

- Provide better environment in terms of modeling which may make complex problems simpler.
- Reducing time taken to solve the problem.
- Combining CP with OR techniques provide better algorithms to detect errors while searching for the optimal solution.
- Using CP techniques will provide better approach to understand OR problems by visualizing the problem structure.

However some disadvantages can arise when integrating CP with OR techniques. These disadvantages are:

- Developing an integrated algorithm may take more time than developing CP algorithm or OR algorithm.
- Integrating both methods may be hard to implement and not easy to understand by others.

Chapter 3

Heuristics and Domain Reduction

In this Chapter we develop a simple greedy search algorithm. The greedy algorithm is used to solve 10 literature benchmark problems. Developing a simple heuristic that is also accurate is a key aim of many researchers. Normally, good VRP heuristic algorithms must meet the following important criteria.

- **Accuracy**

One of the important aspects in the criteria is accuracy since the results obtained by using the heuristic algorithm to solve certain VRPs are essential to decide whether the algorithm is good or bad.

- **Speed**

If the accuracy test decides the good and the bad, ugly algorithms are those taking a long time to find a solution. Speed in solving VRPs is another important point that must be met to provide good heuristic algorithm. Some real-life problems such as pickup and delivery may require fast actions with reasonable accuracy. Getting an accurate solution that takes days to be obtained, may not be considered useful by users who want fast solutions in a dynamic environment.

- **Simplicity**

Easy to understand not hard to code algorithms, are more likely to be used than the more complicated algorithms. The Clark and Wright algorithm stands as clear example of a simple algorithm preferred by end users to solve VRPs over more accurate but more complicated algorithms.

- **Flexibility**

  It's important for any algorithm to be flexible in term of accommodating changes in the input data. Flexibility provides more options to improve the heuristic algorithms.

Section 3.1 provides a simple greedy search algorithm developed by calculating the cost between each edge in order to minimize the overall cost. The greedy search algorithm is implemented and used to solve 10 benchmark capacitated vehicle routing problem instances. Also, in Section 3.1 we apply domain reduction to solve the generated CVRPs using the greedy search algorithm and compare the results.

Section 3.2 observes the effect of the cost or distance matrix on reducing the domain and hence on the obtained results. Four examples are provided to help investigate the role of domain reduction in solving CVRP.

3.1 A Simple Heuristic Algorithm for the Symmetric VRP

Consider the capacitated vehicle routing problem with the following notation:

- $C = \{1, 2, \ldots, n\}$:the set of customer location.
- 0 : depot location.
- $G=(N,E)$: the graph representing the vehicle routing network with $N=\{0,1,\ldots,n\}$ and $E=\{(i,j):i,j \in N,\ i<j\}$.
- q_j: demand of customer j.
- Q : common vehicle capacity.
- m : number of delivery vehicles.
- c_{ij} : cost or distance between locations i and j.
- L : maximum distance a vehicle can travel.
- P_j: a lower bound on the cost of traveling from the depot to customer j.
- $\ell(S)$: lower bound on the number of vehicles required to visit all locations of S in an optimal solution. Note that $S \subseteq C$ and $\ell(S) \geq 1$.
- $\bar{S}$: the complement of S in C

- O: Set of the not selected customers.
- W: Set of selected customers.
- x_{ij}: 1,2, or 0

The requirements are that:

- The total demands for each route must not exceed the capacity of the vehicle.
- All customers must be visited and supplied by exactly one vehicle.

To solve the above CVRPs, we develop a simple heuristic algorithm. The algorithm starts by choosing customers with the lowest distance from the depot. The number of chosen customers is twice the number of the vehicles. Hence, if the number of the routes or vehicles is m, then the algorithm chooses 2m customers with the minimum distance from the depot. Next the algorithm takes the remaining customers one by one and connects them to one of the 2m chosen customers based on the lowest distance and so on until all customers have been chosen. Now the result will be 2m, one way edges from the depot. In order to create m routes, the algorithm connects the last chosen customers based on the lowest cost or distance. This set up provides m routes with a low distance or cost.

However, to check if the set up is a solution, the algorithm calculates the demands for each route and compares it with the capacity. If the set up doesn't violate the capacity constraint, then the set up is a solution to the problem, otherwise a new set up will be done. For the route that violates the capacity the most, the algorithm removes one of the customers (using a removing criterion) and adds the removed customers in the route with minimum demands (using adding criterion). The process will be repeated until all routes demands become less than or equal to the capacity.

The feasible solution obtained by the algorithm will be stored and the algorithm starts searching for another set up that is less than the current solution. The optimizing process will continue until all possible set ups are exhausted. The following describes the greedy search algorithm (Algorithm 1) in detail:

Algorithm 1

Initialization: W = ϕ, O = {1,2,…,n}

Step 1: Choose 2m customers with the lowest distance from the depot, let F = 0, c = common vehicle capacity, d_i is the demand for customer i, O is the set of all non-chosen customers, W is the set of chosen customers Z_n =1000000 (assigning large value to Z_n at start then the value will be updated).

Set up:

Step 2: For each non-chosen customer j from O choose customer i from W such that c_{ij} is the lowest. Update W and O

Step 3: If O = ϕ go to step 4, otherwise go to step 2.

Step 4: For each customer j (the last customer connected) connect the ones with the lowest distance.

Feasibility:

Step 5: Calculate the total distances and demands for each route. If the total demands for each route is less than or equal to the capacity, then go to step11.

Step 6: Choose the route that violates capacity the most. For each customer i in the route (the depot is not included) calculate $b_i = c_{ij} + c_{jk} - c_{ik}$, where i is preceded by customer j (could be the depot) and followed by customer k (could be the depot).

Step 7: Remove customer i with the maximum b_i value and connect customer j with k.

Step 8: Choose the route with lowest total demand. For each customer j and k in the route calculate $a_i = c_{ij} + c_{jk} - c_{ik}$, customer i (i is the customer that had been removed in Step 7) to be added between j and k.

Step 9: Insert customer i between j and k such that a_i is the lowest. F=F+1

Optimizing:

Step 10: If F>3600 stop (this limits the setups to 3600 different ones) otherwise choose different setup and update W and O then go to Step 2.

Step 11: Repeat until all the feasible solutions checked. Let the feasible solution=Z*.

Step 12: If $Z^* \leq Z_n$ then Z_n =Z*

Step 1 is initialization step that assign values to the needed variables. In steps 2 and 3 the algorithm takes the remaining customers one by one and checks the distance between them and the chosen customer. Customers with the lowest distances will be connected and the process will be repeated until all customers are connected. Step 4 decides the group of customers that form a route based on the distance. At this stage the algorithm provides m routes in which all customers are visited by a vehicle. In order to be feasible, the solution must also satisfy the capacity condition that "the total demand for each route must not exceed the capacity of the vehicle". To satisfy this condition, steps 6 to 10 choose the route with total demand that is beyond the capacity the most and also choosing the route with lowest demand. Calculating $b_i = c_{ij} + c_{jk} - c_{ik}$ (b_i is the removing criteria) in the first route and removing the customer with maximum b_i as the equation indicate that removing the customer with the highest b_i will keep the difference in terms of distance. Now to add the removed customer to the lowest demand route while keeping the distance lost to this procedure to a minimum, the algorithm calculates $a_{i^*} = c_{i^*j^*} + c_{j^*k^*} - c_{i^*k^*}$ (a_{i^*} is the adding criteria) and adds the removed customer between the two customers with the lowest a_{i^*}. To avoid repeating steps 5 to 10 without getting a feasible solution, step 9 sets F as a counter to find a feasible solution. The search for feasible solutions will be terminated if the process of removing and adding exceeds 3600 iterations. Steps 11-13 set the obtained feasible solution as Z* and compare it with the value of Z_n as a process to optimize the solution. The process will be repeated until trying all the possible moves and Z_n will be printed as the final solution.

The greedy search algorithm developed in this section can be illustrated by the following flow chart:

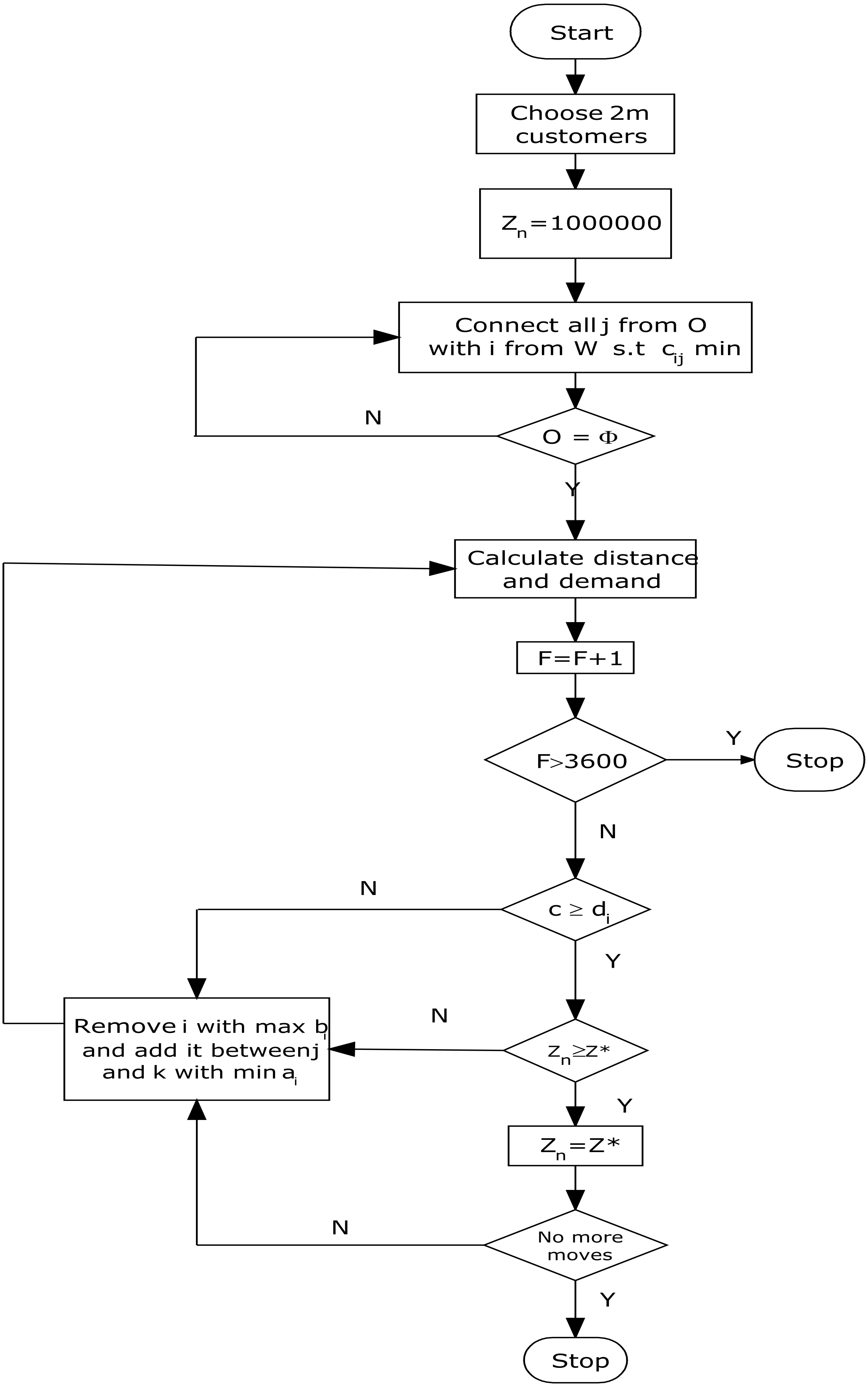


Figure 3.1: Flow Chart for VRP Improved Heuristic Algorithm

We implemented our algorithm in C++ and tested it on 10 literature test problems. The number of customers for the test problems ranged from 7 to 48. The optimal solutions (that we compared our results to) are obtained using CPLEX and the CVRP formulation that mentioned in Section 1.2. Also the Algorithm 1 results are compared to the results obtained by Symphony and the Clarke and Wright Algorithm. Table 3.1 provides details for the benchmark problems.

Problem number	**References**	**Number of customers**
1	Eilon et al (1971)	7
2	Eilon et al (1971)	13
3	Groetschel (1992)	17
4	Groetschel (1992)	21
5	Groetschel (1992)	24
6	Computational Infrastructure for Operations Research 2003	26
7	Computational Infrastructure for Operations Research 2003	29
8	Eilon et al (1971)	31
9	Computational Infrastructure for Operations Research 2003	42
10	Held and Karp (1970)	48

Table 3.1: Benchmark Problems

Table 3.2 provides the computational results for using Algorithm 1 on the above mentioned benchmark problems.

Table 3.2: Algorithm 1 Computation results

Problem number	Optimal		Algorithm 1			Other heuristics			
	Optimal solution	Time in seconds	Solution results	Time in seconds	% from optimal	Symphony solutions	%from optimal	C&W Saving solutions	%from optimal
1	114	23.3	114	0.015	0	114	0	119	4
2	290	2464.73	336	0.001	15.8	300	3	290	0
3	1560	7.20	1909	0.015	22.3	2685	72	2150	38
4	3169	7.15	3833	0.015	21	3704	17	3754	18
5	1373	1002.40	1500	0.015	9	2053	49.5	1659	21
6	1685	275.53	2161	0.015	28	N/A	N/A	1891	12
7	1749	2516.14	2559	0.015	46	2050	17	2107	20
8	1111	18286	1372	0.109	23	N/A	N/A	1336	20
9	1408	18000	2071	0.093	47	1668	18	2391	70
10	13333	18000	21644	0.125	62	14749	11	19342	45

According to Table 3.2, the solution obtained by the algorithm to all the Problems (except 1 and 5) are far from being accurate. We will discuss the reasons that cause this divergence. As Problem 2 is smaller in terms of size we choose to select it and explain the divergence.

Problem 2 is Eilon et al (1971) with 13 customers, 4 trucks, 6000 units capacity, {1200, 1700, 1500, 1400, 1700, 1400, 1200, 1900, 1800, 1600, 1700, 1100} units demands and with distance matrix

$$\begin{pmatrix}
-1 & 9 & 14 & 21 & 23 & 22 & 25 & 32 & 36 & 38 & 42 & 50 & 52 \\
0 & -1 & 5 & 12 & 22 & 21 & 24 & 31 & 35 & 37 & 41 & 49 & 31 \\
0 & 0 & -1 & 7 & 17 & 16 & 23 & 26 & 30 & 6 & 36 & 44 & 46 \\
0 & 0 & 0 & -1 & 10 & 21 & 30 & 27 & 37 & 43 & 31 & 7 & 39 \\
0 & 0 & 0 & 0 & -1 & 19 & 28 & 25 & 35 & 41 & 29 & 31 & 29 \\
0 & 0 & 0 & 0 & 0 & -1 & 9 & 10 & 16 & 22 & 20 & 28 & 30 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 7 & 11 & 13 & 17 & 25 & 27 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 10 & 16 & 10 & 18 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 6 & 6 & 14 & 16 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 12 & 12 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 8 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{pmatrix}$$

Using the modeling and solving language and environment (Xpress mosel) to solve the problem (we assign 1 to depot when using Xpress), we get the following optimal solution with the routes:

<u>Solution</u>

4 routes

Route 1:1- 2-1

Route 2:1- 3-10-9-1

Route 3:1- 5-6-8-7-1

Route 4:1- 11-13-12-4-1

Total distance= 290

While our heuristic gives the solution:

Solution
4 routes
Route 1: 0- 9- 12- 4-0
Route 2: 0- 1- 3- 2- 0
Route 3: 0- 8- 11- 6- 0
Route 4: 0- 5- 7- 10- 0
Total Distance = 336

Comparing the first route in both solutions, one can conclude that any best or optimal solution to the problem must take the first customer alone as a single customer route since the distance between the first customer and the depot is only 9 which gives 18 as the total distance for the first route. This will drop down any solution to the given problem. Unfortunately, our algorithm starts by taking 2m non-removable customers (where m is the number of customers (**Step 1)**) which, means single customer routes solutions are not considered. In the real life problems it's very rare that the solution for a given problem will involve single route customers, as running a vehicle with large capacity to serve only one customer seems unrealistic. For problem 8 the optimal solution takes customer number 30 as a single route customer which makes our solution far for the same reason mentioned above.

3.2 Calculations

Good results can be obtained using greedy search algorithms for VRPs when there is a gap in values between distances in all the rows and/or columns. This gap in values will help the greedy algorithms in finding the feasible solution. Having close values in the row or column that are governed by the demands may provide a solution that is far from the optimal especially in adding and removing customers to meet the capacity constraint.

The search for a feasible solution may lead the algorithm in the direction of choosing big values in order to meet the capacity conditions. The nature of a greedy search algorithm needs differences in values in the distance matrix. Domain reduction requires differences in values so it can eliminate the large distances in the distance matrix. Hence, we can suggest that a greedy search algorithm provides good results for a certain problem as long as the domain of the given problem can be reduced significantly (around 50% from the maximum value given in the distance matrix). If the domain of the problem cannot be reduced significantly from the maximum distance then greedy search algorithm may provide inaccurate solution. To test this we generate 4 distance or cost matrices. Then we solve them using Algorithm 1

Example 1: Consider a CVRP with the following cost or distance matrix.

DISTANCE:

-1	10	20	30	10	20	20	10
0	-1	20	10	10	20	30	20
0	0	-1	30	10	20	15	10
0	0	0	-1	10	20	35	10
0	0	0	0	-1	20	30	15
0	0	0	0	0	-1	30	40
0	0	0	0	0	0	-1	10
0	0	0	0	0	0	0	-1

DEMANDS: [(2) 10 30 10 10 5 5 10]

CAPACITY: 40

Now to reduce the domain significantly we delete the distances within 50% of the maximum distance. In this example we have 40 as the maximum distance or cost, hence all the values above 20 will be deleted. This will provides a distance matrix of the following shape

DISTANCE:

-1	10	20	-	10	20	20	10
0	-1	20	10	10	20	-	20
0	0	-1	-	10	20	15	10
0	0	0	-1	10	20	-	10
0	0	0	0	-1	20	-	15
0	0	0	0	0	-1	-	-
0	0	0	0	0	0	-1	10
0	0	0	0	0	0	0	-1

and solving the resulting problem using Algorithm 1we get:

Solution
2 routes
Route 1: 0- 1- 5- 3- 6- 7- 0
Route 2: 0- 4- 2- 0
Total Distance = 115

Solving the problem without domain reduction using Xpress mosel and fixing 1 as the depot we gets:

Solution
2 routes
Route 1: 1 - 5-3-1
Route 2:1 - 6-2-4-7-8-1
Total distance= 115

Note that the greedy search algorithm found the optimal solution faster than the exact method (Algorithm 1 time is 0.15 seconds and Xpress mosel time is 1.30 seconds). In the next example we change the second row of the distance matrix to closer values.

Example 2: Consider a CVRP with the following cost or distance matrix.

$$\text{DISTANCE:} \begin{pmatrix} -1 & 10 & 20 & 30 & 10 & 20 & 30 & 10 \\ 0 & -1 & 25 & 30 & 25 & 30 & 25 & 30 \\ 0 & 0 & -1 & 30 & 10 & 20 & 30 & 10 \\ 0 & 0 & 0 & -1 & 10 & 20 & 5 & 10 \\ 0 & 0 & 0 & 0 & -1 & 20 & 30 & 10 \\ 0 & 0 & 0 & 0 & 0 & -1 & 30 & 30 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

The maximum distance in this example is 30, hence applying domain reduction within 50% of the maximum distance means deleting all the values above 15. Solving the reduced distance or cost matrix we obtain no feasible. Solving the problem without reducing the domain by 50% will give the following results:

Solution
2 routes
Route 1: 0 -1- 6- 7- 0
Route 2: 0- 2- 4 -3 -5- 0
Total Distance = 135

Solving the same problem using Xpress mosel and assigning 1 to the depot we get:

Solution
2 routes
Route1:1- 5-4-7-2-1
Route2:1 - 8-3-6-1
Total distance=120

The result obtained by the greedy search algorithm exceeds the 10% from the optimal solution. For this problem the greedy search algorithm may not be the best choice. The domain reduction for the problem indicates that the values in the distance matrix are so close it also reveals that the simple greedy search algorithm to deal with the problem may not be a good choice.

To investigate the effect of domain reduction more we generate an 18x18 matrix in the next example.

Example 3: Consider a CVRP with the following cost or distance matrix.

DISTANCE:

-1	121	518	142	84	297	35	29	36	236	390	238	301	55	96	153	336	111
0	-1	246	745	472	237	528	364	332	349	202	685	542	157	289	426	483	155
0	0	-1	268	420	53	239	199	123	207	165	383	240	140	448	202	57	200
0	0	0	-1	211	466	74	182	243	105	150	108	326	336	184	391	145	40
0	0	0	0	-1	70	567	191	27	346	83	47	68	189	439	287	254	250
0	0	0	0	0	-1	324	638	437	240	421	329	297	314	95	578	435	300
0	0	0	0	0	0	-1	353	282	110	324	61	208	292	250	352	154	170
0	0	0	0	0	0	0	-1	505	289	262	476	196	360	444	402	495	120
0	0	0	0	0	0	0	0	-1	259	555	372	175	338	264	232	249	70
0	0	0	0	0	0	0	0	0	-1	134	530	154	105	309	34	29	45
0	0	0	0	0	0	0	0	0	0	-1	80	572	196	77	351	63	89
0	0	0	0	0	0	0	0	0	0	0	-1	150	488	112	120	267	316
0	0	0	0	0	0	0	0	0	0	0	0	-1	412	227	169	383	20
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	91	661	228	117
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	257	390	42
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	633	31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	215
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: [0 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 30]

CAPACITY: 70.

Solving the problem using the greedy search algorithm and reducing the domain by 50% we get:

Solution

3 routes

Route 1: 0- 6 -11 -10- 16- 4- 12- 0

Route 2: 0- 3- 9- 15- 17- 7- 0

Route 3: 0- 8- 2- 5- 14- 13- 1- 0

Total Distance = 1999

Now solving the same problem in order to find the optimal solution we get:

Solution

3 routes

Route 1: 1-2-11-12-7-1

Route 2: 1-4-8-18-13-5-9-1

Route 3: 1-16-10-17-3-6-15-14-1

Total distance= 1957

Note that the domain of the problem is reducible by 50% from the maximum value given in the distance matrix and the result obtained by the heuristic algorithm is very close to the optimal (only 2% from the optimal).

Example 4:

Changing the last row/column in the distance matrix in Example 3 from

111 155 200 40 250 300 170 120 70 45 89 316 20 117 42 31 215 0

to

390 399 393 400 399 396 397 390 395 410 389 392 410 395 400 399 390 0

we have close values to the maximum distance given in the distance matrix. Now solving the new modified problem using the heuristic algorithm without reducing the domain (since no feasible solution can be obtained if we reduce the domain by 50%) we obtain:

Solution

3 routes

Route 1: 0- 7- 17- 10- 14- 13- 0

Route 2: 0- 3- 16- 9- 15- 12- 8- 0

Route 3: 0- 6- 11- 4- 5- 2- 1- 0

Total Distance = 2394

Now solving the same problem using Xpress to find the optimal solution we get:

Solution
3 routes
Route 1: 1 - 2-18-8-1
Route 2: 1 - 7-4-11-12-13-5-9-1
Route 3: 1 - 14-15-6-3-17-10-16-1
Total distance= 2126

It's clear that the solution obtained by the heuristic algorithm is more than 10% from the optimal. The following table provides more details:

Example number	Optimal results	Greedy search results	Results with domain reduced by 50%	Percentage from optimal
1	115	115	115	0%
2	120	135	N/A	12%
3	1957	1999	1999	2%
4	2126	2394	N/A	13%

Table 3.3 Domain reduction results

3.3 Conclusion

Table 3.3 illustrates that if the distance matrix of a VRP instance cannot be reduced significantly then the results obtained by the greedy search algorithm may not be accurate. As we observed, greedy search algorithms may provide more accurate results if applied to solve VRP instances that allow a significant domain reduction. According to the examples in this Chapter the form of the given data matrix influences not only the size of the problem, but also how hard the problem is. Although it's simple, fast and flexible, the accuracy of the greedy search algorithm that we developed in this Chapter may require some improvement. Observing the effect of domain reduction on the generated problems, we will combine in the next Chapter the greedy algorithm with domain reduction and observe the results.

Chapter 4

Heuristic Algorithm for CVRP

VRP heuristic algorithms can be divided into two types: Classical heuristics such as: the Clark and Wright algorithm (1964), the sweep algorithms and the Fisher and Jaikumar (1981) algorithm, and metaheuristics such as: Simulating Annealing and Genetic algorithms. Heuristic algorithms have proved to be very useful for solving large VRPs in reasonable time. Also, good heuristics can provide good upper bounds that play an important role in exact methods.

This Chapter provides computational results that show the domain reduction can improves the Clarke and Wright algorithm by 8% and Algorithm 1 by 24% when combined with **Distance Constrained VRP (DCVRP)**. Also, the Chapter investigates the effect of domain reduction on Simulating Annealing metaheuristic.

In Section 4.1 we provide a description to the domain reduction restriction that we will use in this Chapter. Section 4.1.1 combines the domain reduction condition with the greedy search algorithm that we described in Chapter 3 (Algorithm 1). Section 4.1 discuses the importance of tightening Algorithm 1 and we propose a **Distance Constrained VRP (DCVRP)** as an approach. Section 4.1.2 describes (DCVRP), and provides the mathematical formulation to the problem. Section 4.1.2 Also provides computational results for using Algorithm 2 (a combination of Algorithm 1, domain reduction and DCVRP) to solve the 10 benchmark problems that we mentioned in Chapter 3.

Section 4.2 combines the Clarke and Wright (C&W) algorithm with the domain reduction to solve the 10 literature benchmark CVRPs.

Section 4.3 describes Zbigniew and Piotr (2002) Simulating Annealing (SA) algorithm and uses it to solve the 10 benchmark CVRPs. This Section observes that the domain reduction didn't affect the results of Simulating Annealing metaheuristic (SA) when applied to solve the 10 benchmark CVRPs.

Section 4.4 uses Algorithm 2, (C&W) and (SA) to solve large VRPs combined with domain reduction. The obtained results showed that combining domain reduction with the Clarke and Wright algorithm improve the results by 39% when applied to large CVRP instances. Section 4.5 concludes the Chapter.

4.1 Domain Reduction

To survey the influence of domain reduction on our solution we added a new constraint that deletes some large numbers from the distance matrix and thus forbids the use of certain links. The new restriction is

$$c_{ij} \leq R \qquad\qquad i,j=1,2,\ldots,n$$

where c_{ij} represent the cost between i and j, and R is a threshold that depends on the maximum number in the distance matrix.

The new domain reduction restriction will delete some unneeded values from the distance matrix and setting the components to “0”. This may help tighten our heuristic and change the direction of the search.

4.1.1 Computations

In order to observe the effect of the domain reduction restriction more closely, the value of R will be determined manually by the user based on the maximum number in the distance matrix. The way we implement the algorithm will calculate the maximum distance used in the distance matrix and the program will not start unless we give a percentage on how far from the maximum we need the value of R. If we take Problem 2 as an example we can see that the maximum distance used in this problem is 128. By directing the program to solve Problem 2 and assigning 0 to the percentage, the program will take 100% of the maximum distance. Hence, 90 means the program set the values above 90% of 128 to infinity.

Algorithm 1 showed some weakness when removing and adding the nodes from the violating routes. In Algorithm 1 removing nodes one by one to meet the capacity can increase the objective value rapidly especially when dealing with hard VRPs. One can suggest removing two or more nodes to improve the solution. However by removing two or more customers every time, we may lose the simplicity and the speed gained by our developed algorithm.

In Algorithm 1, we use the procedure of removing and adding customers from the routes without any restrictions on the distance. Using simple equations (removing equation) $b_i = c_{ij} + c_{jk} - c_{ik}$ and (adding equation) $a_{i^*} = c_{i^*j^*} + c_{j^*k^*} - c_{i^*k^*}$ only will direct the search after the initial setup to focus on meeting the capacity constraint without a real restrictions on how far it can increase the distance in the process.

In order to tighten the solution, the distance constraint vehicle routing problem (DCVRP) may be helpful. The restrictions that (DCVRP) applied on each route may be useful in directing the removal and adding customers from each route combined with domain reduction.

A combination of the greedy search algorithm (Algorithm 1), domain reduction and distance restriction on each route will be presented next, but first we will give a brief definition to distance constraint vehicle routing problem (DCVRP) and describe some of the theory and computations.

4.1.2 Distance Constrained Vehicle Routing Problem (DCVRP)

The distance constrained vehicle routing problem (DCVRP) is another variant of VRP. The problem is similar to CVRP with extra condition; the total distance (time) traveled by each vehicle must not be more than a pre-specified number. i.e the (DCVRP) objective is to minimize the cost or the total distance traveled by the vehicles without violating the following restrictions:

(a) The demands of all customers must be met.

(b) The capacity of vehicles may not be violated (i.e. for each route the total demands must not exceed the vehicle capacity).

(c) The total time (or alternatively distance) for each vehicle to complete its tour may not exceed some predetermined level. Referring to Laporte, Desrochers and Nobert (1984), the mathematical formulation for the problem is:

$$\text{minimize } Z=\sum\sum c_{ij}x_{ij} \quad i\in N,\ i<j \tag{4.3.1}$$

subject to

$$\sum x_{0i}=2m \quad i\in N \tag{4.3.2}$$

$$\sum x_{ij}+\sum x_{ji}=2 \quad j<i \text{ or } i<j,\ i\in N \tag{4.3.3}$$

$$\sum x_{ij}\le |S|-\ell(S), \quad i,j\in S, \quad S\subseteq N, 3\le |S|\le n-2 \tag{4.3.4}$$

$$x_{ij}=1,2,\text{or } 0 \tag{4.3.5}$$

$$m \text{ is a positive integer} \tag{4.3.6}$$

where

- N= {1, 2,…, n}:the set of customer location.
- 0 : depot location.
- G=(N,E) : the graph representing the vehicle routing network with N={0,1,…,n} and E={(i,j):i,j$\in$N, i<j}.
- q_i : demand of customer j.
- Q : common vehicle capacity.
- m : number of delivery vehicles.
- x_{ij} :distance between locations i and j.
- L : maximum distance a vehicle can travel.
- P_j: a lower bound on the cost of traveling from the depot to customer j.
- $\ell(S)$: lower bound on the number of vehicles required to visit all locations in S

In our implementation for the new algorithm, we specify the value of R as an addition to Algorithm 1. R is to be determined based on the largest distance or cost value in the distance (cost) matrix. The resulting algorithm will be referred to as Algorithm 2. R will be used as threshold in order to direct the search. The restrictions on each route will be selected in a way that tighten the search and less than the value of L. Applying the algorithm to solve the previously mentioned 10 problems and using the domain reduction and distance restriction we get the following results.

Problem number	Optimal solution	Algorithm 2		Other heuristics				Max value	distance	Domain reduces
		solution	% from optimal	Symphony solution	%from optimal	Saving solution	%from optimal			
1	114	114	0	114	0	119	4			
2	290	336	15.8	300	3.4	290	0	128	0	0
		298	2.7						105	80%
		314	8						100	0
		N/A	N/A						100	80%
3	1560	1909	22.3	2685	72	2150	38	717	0	0
		N/A	N/A						600	0
		2413	N/A						700	0
		1881	20						900	80%
		1719	10						1010	75%
4	3169	3833	21	3704	17	3754	18	1611	0	0
		3837	22						1500	70%
		3755	18						1400	70%
		3639	15						1390	80%
5	1373	1500	9	2053	49.5	1659	21	516	0	0
		1750	27						500	0
		1651	20						500	40%

Table 4.1a: Domain Reduction Computation and DCVRP Results

Problem number	Optimal solution	Algorithm 2		Other heuristics				Max value	distance	Domain reduces
		solution	% from optimal	Symphony solution	%from optimal	Saving solution	%from optimal			
6	1685	2161 2037 2004 1911	28 21 19 13	N/A	N/A	1891	12	925	0 900 800 700	0 80% 60% 70%
7	1749	2559 2326 2066	46 33 18	2050	17	2107	20	821	0 800 750	0 60% 60%
8	1111	1372 1389	23 24	N/A	N/A	1336	20	229 223	0 300	0 90%
9	1408	2071 1823 1802 1790	47 29 28 27	1668	18	2391	70	599	0 550 490 490	0 60% 80% 50%
10	13333	21644 21077 20137 19197 14209	62 60 51 44 7	14749	11	19342	45	6571	0 6500 5500 5800 4000	0 50% 60% 40% 60%

Table 4.1b: Domain Reduction Computation and DCVRP Results

From Table 4.1(a and b), we conclude that the domain reduction improves the costs rapidly. Algorithm 2 is far better than Algorithm 1 in terms of accuracy.

4.2 Clarke and Wright (C&W) Algorithm

This section combines the domain reduction with Clarke and Wright algorithm. The algorithm applied to solve the 10 benchmark VRP instances.

Problem number	Optimal solution	Modified C&W	%from optimal	Domain reduced
1	114	119	4	N/A
2	290	290	0	N/A
3	1560	2150	38	N/A
4	3169	3754 3658	18 15.4	0 62%
5	1373	1659 1579 1404	21 15 2.3	0 65% 70%
6	1685	1891 1888	12.2 12	0 50%
7	1749	2107	20	0
8	1111	1336 1278	20 15	0 5%
9	1408	2391 1999 1747	70 42 24	0 50% 55%
10	13333	19342 19181	45 43.8	0 65%

Table 4.2: The C&W Saving Algorithm and Domain Reduction

Table 4.2 provides clear results on how the domain reduction can minimize the cost when combined with the Clarke and Wright algorithm.

Combining the domain reduction with the classical heuristics will improve the solution, as detailed in tables 4.1 and 4.2.

The next section will investigate the effect of domain reduction on one of the metaheuristics.

4.3 Simulating Annealing Algorithm (SA)

To investigate the effect of domain reduction when combined with a metaheuristic algorithm, this section presents one of the simulating annealing algorithms. The algorithm uses the annealing temperature T developed by Zbigniew, and Piotr (2002) and the greedy search algorithm developed in Chapter 3 (Algorithm 1). The SA algorithm can be described in the following steps:

Step 1: Using Algorithm 1, find initial solution.

Step 2: Calculate T= γ *(d+ σ (cn+ e_{min})), where γ <1, d is the total travel distance of the routes, σ is a constant(fixed to 1), c is the number of vehicles, n is the number of customers, and e_{min} is the number of customers in the shortest route. Set f=0. f is a counter.

Step 3: Set f=f+1.

Step 4: Repeat n^2 times, swap 2 customers in each route. Store the new route if it's better than the original.

Step 5: If T<f then print the best solution and stop, otherwise go to step 6.

Step 6: Take a "snapshot" to the initial solution and generate another one using Algorithm 1 and go to step 2.

The restriction

$$c_{ij} \leq R \quad i,j=1,2,\ldots,n.$$

is added as a domain reduction condition. The SA algorithm will calculate the maximum distance used in the distance matrix and let the user choose a percentage on how far from the maximum the value of R wanted. Implementing the SA algorithm and domain reduction using C++ we get the following results:

Problem number	Optimal solution	Modified SA Algorithm		Other heuristics				Domain reduces
		Results	% from optimal	Symphony results	%from optimal	C&W Saving results	%from optimal	
1	114	114	0	114	0	119	4	0
2	290	290	0	300	3.4	290	0	0
3	1560	1629 1686 1700	4.4 8 9	2685	72	2150	38	0 80% 60%
4	3169	3314 3463 3494	4.5 9.2 10.2	3704	17	3754	18	0 80% 60%
5	1373	1473 1431 1545	7.2 4.2 12.5	2053	49.5	1659	21	0 80% 60%
6	1685	1779 1704 1715	5.5 1.1 1.7	N/A	N/A	1891	12	0 80% 60%
7	1749	1945 2131 2022	11.2 22 15.6	2050	17	2107	20	0 80% 60%
8	1111	1269 1349	14.2 21.4	N/A	N/A	1336	20	0 80%
9	1408	1528 1599 1562	8.5 13.5 10.9	1668	18	2391	70	0 80% 60%
10	13333	17888 18391 18302	34 37.9 37.2	14749	11	19342	45	0 80% 60%

Table 4.3: SA and Domain Reduction

Unlike the classical heuristics, metaheuristics combined with domain reduction may increase the cost. Domain reduction seems to work perfectly when combined with a classical heuristic algorithm, but fail to improve the solution when combined with the metaheuristics.

4.4 Heuristics and large instances

Besides providing upper bounds, heuristics are normally useful whenever exact algorithms fail. In most of the cases, exact algorithms face a real challenge when applied to solve large VRP instances in terms of the time and space required to solve the problem to optimality. Also, heuristics can deal with large VRPs efficiently in terms of time taken to solve the problem.

In order to investigate the effect of domain reduction on the large VRPs, we applied Algorithm 2, the Clarke and Wright algorithm and the SA algorithm to 4 large instances. The set of instances are from Christofides, Mingozzi, and Toth, (1979). The details of each instance and the best published solution can be found at Computational Infrastructure for Operations Research (2003).Table 4.4 shows the results:

Dimension	Modified C&W	Modified SA	Algorithm 2	SYMPHONY	Domain reduced %
101	803.439	409.918	1590	820	0
	726.249	409.918	1590	N/A	30
	672.280	409.918	1590	N/A	25
121	933.738	336.485	1401	1034	0
	573.689	336.485	1401	N/A	30
151	958.464	368.996	1498	1053	0
	894.140	368.996	1498	N/A	30
	480.326	368.996	1498	N/A	45
200	1290.961	652.158	1975	1373	0
	1079.164	652.158	1975	N/A	35
	696.730	652.158	1975	N/A	45

Table 4.4: Heuristics and Large VRPs

From Table 4.4, we can observe that the domain reduction reduced the cost significantly when combined with the Clarke and Wright algorithm. For the problem of dimension 101 customers, domain reduction improved the solution by 16%. For the second problem (121 customers) the solution has been improved by 38%. For the third large problem with dimension 151 customers, the solution has been improved by 49.8%. The solution for the problem of dimension 200 customers has been improved by 46%. From Table 4.2 and 4.4 we observe that the domain reduction combined with Clarke and Wright improves the solution rapidly as the size of the problems become large. In addition neither SA nor Algorithm 2 shows any significant response in term of reducing the cost when combined with domain reduction to solve large scale VRPs.

4.5 Conclusion

The results obtained by combining domain reduction with distance restrictions shown in Table 4.1 are good considering the time to solve each problem (the overall time is 0.45 second). The greedy search algorithm provides good results when domain reduction and distance restrictions for each route get involved in directing the search. Another thing that can be concluded is the rapid improvement for problems 9 and 10 in terms of the cost. Also, domain reduction improves the cost when combined with the Clarke and Wright savings algorithm. This improvement can be seen clearly in Table 4.2 especially problem 9, as the cost decreases from 70% from the optimal to 24%. However the results obtained by SA are far better than those obtained by Algorithm 2 and the saving algorithm. Reducing the domain minimizes the cost significantly in Algorithm 2 and the Savings Algorithm, but fails to improve the solution when combined with SA. The deep search procedure for SA provides the first result as the best obtained. Deleting values from the domain didn't help improving the solution for SA algorithm. We observed that SA algorithm is better than Algorithm 2 and the savings algorithm in terms of accuracy. However, the classical algorithms are easy to understand and take less time to be implemented. Furthermore when dealing with large scale VRPs the Clarke and Wright saving algorithm shows an outstanding improvement when combined with domain reduction. From Table 4.4 we can observe that the obtained solution in each case decreased significantly when we apply the Clarke and Wright saving algorithm with domain reduction.

After we explore the effect of domain reduction on solving vehicle routing problem using heuristic methods, the next chapter will apply an exact method to solve VRP combined with domain reduction and observe the effect of reducing the domain on the time taken and gap closing.

Chapter 5

A hybrid Method to solve VRP

In this Chapter we consider the capacitated vehicle routing problem. The branch and cut procedure is used to solve the 10 benchmark problems without applying the domain reduction constraint, analyzing the results then solving the same problems after adding the domain reduction constraint and comparing the results. The computational results provided in this Chapter show that branch and cut combined with the domain reduction can improve the time taken to solve the problem by 48% in comparison with using branch and cut only. In most of the cases the solution value will remain the same. However, in some problems the solution may become slightly higher but the improved significantly.

Section 5.1 describes the implementation of the domain reduction restriction. Section 5.2 details how we combine domain reduction with the branch and cut (exact) method. This Section illustrates the effect of domain reduction in reducing the duality gap (the difference between primal and dual objective values) when combine with branch and cut method. Also, this Section shows the effect of domain reduction on the time taken to solve VRPs. Section 5.3 concludes this chapter.

5.1 Domain Reduction condition and Implementation

The distance matrix for VRP represents the problem domain. Hence, to reduce the domain we must reduce the domain by eliminating some numbers from the distance matrix. As described earlier a simple restriction developed to reduce the domain can be described mathematically as

$$c_{ij} \leq R \qquad \text{for all i and j}$$

where c_{ij} is the cost or the distance between node i and j, and R is a threshold chosen logically. Furthermore R value depends deeply on the maximum cost (distance) in the cost matrix.

As we mentioned earlier this thesis focuses on the Symmetric Capacitated Vehicle Routing Problem (CVRP) with single commodity and one depot. The restrictions are capacity and cost or distance. Moreover, as we are dealing with exact method in this Chapter we expect the improvement of combining domain reduction will apply to time taken only.

5.2 Calculation

We considered the CVRP formulation provided in Section 2.5.1. We use CPLEX (ILOG SA) to solve the ten instances used in Chapter 4. We will combine the branch and cut method with the domain reduction constraint, starting from a distance close to the maximum cost (distance) down until we reach a value for which a feasible solution cannot be found. We will analyze the results in each case in terms of time and the gap closure in order to reach an understanding of the effect of the domain reduction on the exact methods.

For each problem we find the maximum distance in the distance matrix and flag it as a threshold, then eliminate all the distances above a chosen percentage from the maximum. We decreased the percentage gradually until no initial feasible solution can be found. The values of R, duality gaps, optimal solutions and the time taken to solve each problem will be presented next but first we will highlight the influence of domain reduction on closing the duality gap.

Recall the 10 benchmark problem mentioned in the previous Chapters. Problem 9 was chosen to illustrate the effect of the domain reduction on VRPs.

- **Problem 9 (42 customers):** We choose this problem to show the effect of domain reduction on the duality gap. Problem 9 is one of the hard literature problems that require a long time to be solved optimally. In addition, the initial duality gap for problem 9 is almost 50%. For this reason, the problem is useful for illustrating the effect of domain reduction on the duality gap.

Solving problem 9 using branch and cut only and without reducing the domain we get:

Objective	Gap	Depth	CPU Time (Sec)
1429.2 1459 1483.2 1490.25 1462.1	49.75% 49.75% 49.75% 49.75% 49.75%	93 192 292 392 492	0
1449.7485 1429.1009 1446.4211 1382.4805 1456.9	49.75% 49.75% 49.75% 49.75% 49.75%	1092 1192 1292 1392 1492	49.19
1431.5 1447.45 1427 1499.1667	49.75% 49.75% 49.34% 49.34%	2092 2191 14 105	80.06
1477.5405 1434 1402.5 cutoff 1498.1197 1499	49.75% 49.75% 49.34% 49.34% 49.34%	2092 2191 14 205 305	104.39
. . . . .	. . . . .	. . . . .	. . . . .
1408	3.70%	14	18000

Table 5.1: Duality Gap and First Domain Reduction

When reducing the domain by 80% from the maximum value used in the distance matrix we get:

Objective	Gap	Depth	CPU Time (Sec)
			0
1414.5000	18.54%	89	
1425.9268	18.11%	189	
1426.5139	17.81%	278	
1474.6667	17.81%	84	
1484.0132	17.81%	175	
			58.88
1429.2000	15.78%	46	
1459.0000	15.78%	141	
1483.2000	15.78%	236	
1490.2500	15.58%	70	
0	13.51%	82	
1462.1000	13.51%	102	
1449.7485	13.37%		
			121.81
1431.5000	13.19%	74	
1447.4500	13.19%	171	
1427.0000	13.19%	64	
1299.1667	13.17%	10	
0	10.42%	49	
1377.5405	10.34%	21	
1334.0000	10.17%	17	
1402.5000	10.13%	34	
. . . .	. . . .	. . . .	. . . .
1408	1.12%	12	18000

Table 5.2: Duality Gap and Second Domain Reduction

Note that the initial gap reduced from 49.75% to 18.54%, when the domain reduced by 80% from the maximum distance in the distance matrix. Also when solving the problem without the domain reduction, the gap was 49.34% after about 105 seconds. When the domain reduced by 20%, the gap was about 10.34% (after 105 seconds). Furthermore, when reducing the domain by 60% from the maximum value used in the distance matrix we get:

Objective	Gap	Depth	CPU Time (Sec)
			0
1415.0600	20.12%	96	
1427.6250	20.12%	193	
1429.1224	20.12%	293	
1429.8421	20.12%	393	
1430.7692	20.12%	493	
			31.69
1495.4000	20.12%	1087	
1326.0000	18.38%	5	
1429.7857	18.38%	99	
1445.7449	13.50%	291	
1476.8571	13.50%	25	
			54.97
1422.4444	13.33%	210	
1442.8030	13.33%	310	
1465.8750	13.33%	410	
1492.2727	13.33%	510	
1406.1870	11.51%	184	
			106.49
1411.0000	5.93%	45	
1394.8750	5.89%	22	
1401.0506	5.82%	88	
1405.6733	5.82%	185	
1411.2500	5.79%	45	
. . . . .	. . . . .	. . . . .	. . . . .
1408	0		13056

Table 5.3: Duality Gap and Third Domain Reduction

Note that, although the initial gap (20.12%) when reducing the domain by 40% is not as good as the initial gap obtained by reducing the domain by 20% (18.54%), the gap after 105 seconds for the third result was about 5.45% which is better than the 10.34% obtained by reducing the domain 20% and after the same time. In addition, when reducing the domain by 40% from the maximum value used in the distance matrix we get:

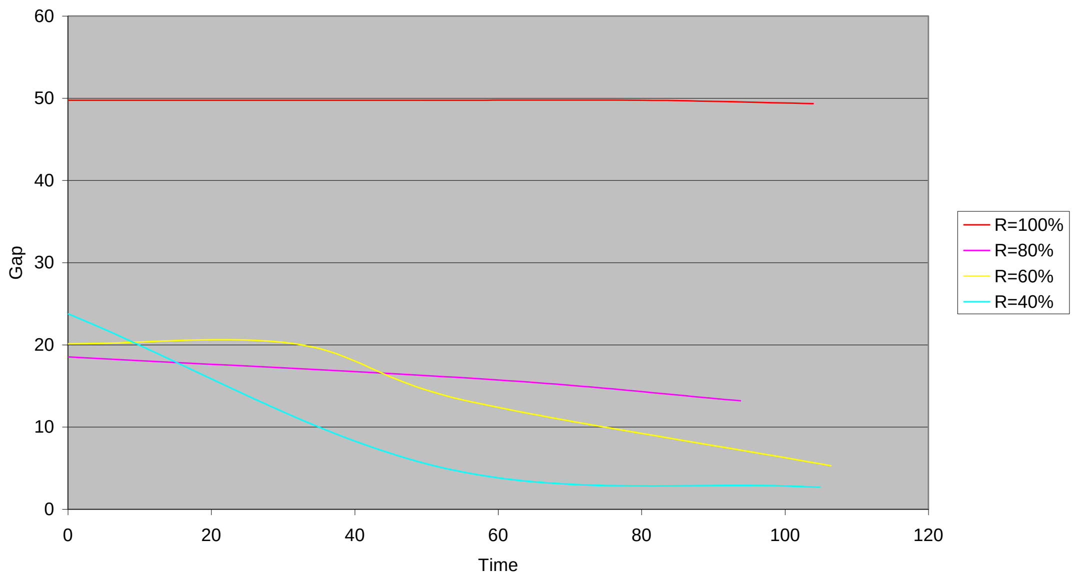
Objective	Gap	Depth	CPU Time (Sec)
			0
1411.9000	23.78%	98	
1417.7692	8.09%	198	
1424.0500	8.04%	64	
1397.3333	7.02%	22	
1418.7632	7.02%	122	
1409.2917	7.02%	48	
1416.4167	7.02%	24	
			23.59
1399.5000	5.15%	15	
1390.0833	5.15%	43	
1409.3636	5.15%	11	
1372.7000	5.13%	25	
1420.6667	5.08%	30	
1421.8190	5.08%	37	
			52.19
1415.4167	2.75%	20	
1402.8500	2.74%	24	
1413.5341	2.73%	28	
1412.8128	2.72%	23	
1405.0500	2.71%	19	
1383.6346	2.69%	16	
. . . .	. . . .	. . . .	. . . .
1417	0		2769.90

Table 5.4: Duality Gap and Fourth Domain Reduction

Although the initial gap (23.78%) when reducing the domain by 60% is not as good as the initial gap obtained by reducing the domain by 20% (18.54%) or when reducing the domain by 40% (20.12%), the gap after 105 seconds for the fourth result (2.68%) was far better than the other results after the same time. Also, reducing the domain by 60% made it possible to find the solution after 2769.90 seconds. However, the obtained solution (1417) after reducing the domain by 60% is not as good as previous ones (1408).

Figure 5.1, illustrates the effect of domain reduction on the gap (Note that the time units are seconds).

Figure 5.1 Duality gap and domain reduction



After showing the effect of domain reduction on closing the duality gap, the following table provides detailed results when applying branch and cut combined with domain reduction to solve the previously mentioned 10 VRPs.

Problem number	CPU Time /second	Solution	Duality gap %	Initial gap %	Eliminated columns / rows	R %
1	23.3	114	0	29.91	0/0	100
	20.90	114	0	27.59	8/12	90
	26.80	114	0	33.33	24/38	70
	18.14	114	0	29.82	40/64	50
2	2464.73	292	0	59.59	0/0	100
	1526.42	292	0	49.75	52/84	80
	2878.95	292	0	40.00	104/164	60
	355.16	298	0	38.89	144/228	40
3	7.20	1560	0	28.11	0/0	100
	6.25	1560	0	21.51	204/309	80
	10.60	1560	0	16.50	222/336	60
	8.40	1560	0	26.36	267/405	40
4	7.15	3169	0	26.57	0/0	100
	11.15	3169	0	24.53	12/18	80
	14.34	3169	0	31.39	54/81	60
	5.33	3169	0	7.91	663/1008	40
5	1002.40	1373	0	24.07	0/0	100
	1258.78	1373	1.58	20.	48/72	90
	1051.66	1373	2.26	2465	26/404	80
	541.42	1373	0	19.04	1324/2008	60
	124.64	1459	0	49.96	1736/2632	40
				23.77		
6	275.53	1685	0	16.04	0/0	100
	103.45	1685	0	10.82	42/63	90
	582.23	1685	0	8.13	84/126	80
	83.95	1685	0	10.16	456/696	60
	9.94	1750	0	8.67	690/1053	40
7	2516.14	1749	0	68.78	0/0	100
	721.75	1749	0	22.87	112/168	80
	1224.00	1749	0	59.52	672/1012	60
	1817	1749	0	14.23	1544/2336	40
8	18286.00	1111	7.26	39.09	0/0	100
	18286.00	1111	6.57	24.70	90/182	90
	18286.00	1118	7.98	29.85	133/259	80
	Infeasible	Infeasible				60
9	18000	1408	3.70	23.10	0/0	100
	18000	1408	1.12	18.54	384/588	80
	13056	1408	0	20.12	2776/4216	60
	2769.90	1417	0	23.78	5196/7864	40
10	18290	13333	3.42	28.10	0/0	100
	12653	13333	3.09	77.83	96/144	80
	7160	13333	3.28	31.22	376/564	60
		Infeasible				40

Table 5.5: Using Exact Method and Domain Reduction to Solve VRPs

5.3 Conclusions

The results obtained by using branch and cut and domain reduction illustrate the importance of domain reduction in reducing the time taken to solve the problems and reducing the duality gap. In some problems the time and the duality gap reduced rapidly but the solution was slightly above the optimal. Also in some cases reducing the domain may increase the time. However, a good results obtained when the domain had been reduced by around 60% from the maximum value in the distance matrix (except in the case of 31 customers). Table 5.5 illustrates clearly that domain reduction reduces the time taken to solve CVRP when combine with the branch and cut exact method.

Chapter 6

Conclusions and Future Work

The Vehicle Routing Problem VRP is different from almost all other optimization problems. The importance of VRP in reducing the cost of any distribution network that involves transportation as well as providing good customer service (by satisfying customer demands), forced the formulation of the problem to find the balance between reducing the cost and satisfying customer demands. Hence, the equation of cost demand capacity made CVRP complicated and extremely hard as the dimensions of the problem increases.

For a long time, simple heuristics have failed to provide satisfactory solutions when applied to VRP as we also found in Chapter 3. However, by reducing the domain and force route restrictions, a simple greedy search algorithm performs better. Deleting some values from the domain may help in some instances, but in general it may direct the search to the wrong area especially if the heuristic algorithm depends closely on choosing the next low value in the domain to form a route. As a result, applying route restrictions helped directing the search. Using domain reduction and applying restrictions on each route improves the greedy algorithm by 24% as we see in Chapter 4. Also, Chapter 4 provides computational results that illustrate clearly the effect of domain reduction when combined with the Clarke and Wright algorithm. The Clarke and Wright algorithm has been improved by 8% when combined with domain reduction.

Chapter 5 combined branch and cut with the domain reduction. The CPU time taken to solve the problems has been reduced by 49.8% when domain reduction is applied.

In general, the results obtained by combining domain reduction with heuristics and exact methods were significant and encouraging. A future work can be highlighted in the next Section

6.1 Future Work

The pruning that constraint programming provides is a huge encouragement to explore more CP techniques. One of the techniques that need to be explored is constraint propagation. As we mentioned in Chapter 2, to develop a constraint propagation algorithm one of the following approaches must be followed:

- **Backtracking Search**

  The method is a combination of Arc consistency and Backtracking; it starts by guessing solutions then test the guessed solution for Arc consistency.

- **Forward Checking**

  This method uses restricted arc consistency between the current variable and the future variables.

- **Look Ahead Search**

  Unlike forward checking, this method doesn't look for restricted arc consistency between the current variable and the future variables only but also performs full arc consistency search.

Note that developing a hybrid approach that combines constraint propagation with OR methods to solve CVRP must overcome the problem of chronological backtracking (that all decisions must be undone in the reverse of the order they were made). Finding the right approach to combine constraint propagation with OR methods to solve CVRP seems interesting as well as challenging for the future work.

Appendix A

EXAMPLE 1- 18 customers generated matrix
CAPACITY : 70

121 518 142 84 297 35 29 36 236 390 238 301 55 96 153 336 111 246 745 472 237 528 364 332 349 202 685 542 157289 426 483 155 268 420 53 239 199 123 207 165 383 240 140 448 202 57 200 211 466 74 182 243 105 150 108 326 336 184 391 145 40 70 567 191 27 346 83 47 68 189 439 287 254 250 324 638 437 240 421 329 297 314 95 578 435 300 353 282 110 324 61 208 292 250 352 154 170 505 289 262 476 196 360 444 402 495 120 259 555 372 175 338 264 232 249 70 134 530 154 105 309 34 29 45 80 572 196 77 351 63 89 150 488 112 120 267 316 412 227 169 383 20 91 661 228 117 257 390 42 633 31 215

DEMAND : 0 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 30

EXAMPLE 2- 7 customers Eilon, Watson-Gandy and Christofides (1971)
CAPACITY : 3

$$\begin{pmatrix} -1 & 10 & 20 & 25 & 25 & 20 & 10 \\ 0 & -1 & 12 & 20 & 25 & 30 & 20 \\ 0 & 0 & -1 & 10 & 11 & 22 & 30 \\ 0 & 0 & 0 & -1 & 2 & 11 & 25 \\ 0 & 0 & 0 & 0 & -1 & 10 & 20 \\ 0 & 0 & 0 & 0 & 0 & -1 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

DEMAND: 0 1 1 1 1 1 1

EXAMPLE 3-13 customers Eilon, Watson-Gandy and Christofides (1971)
CAPACITY : 6000

-1	9	14	21	23	22	25	32	36	38	42	50	52
0	-1	5	12	22	21	24	31	35	37	41	49	51
0	0	-1	7	17	16	23	26	30	36	36	44	46
0	0	0	-1	10	21	30	27	37	43	31	37	39
0	0	0	0	-1	19	28	25	35	41	29	31	29
0	0	0	0	0	-1	9	10	16	22	20	28	30
0	0	0	0	0	0	-1	7	11	13	17	25	27
0	0	0	0	0	0	0	-1	10	16	10	18	20
0	0	0	0	0	0	0	0	-1	6	6	14	16
0	0	0	0	0	0	0	0	0	-1	12	12	20
0	0	0	0	0	0	0	0	0	0	-1	8	10
0	0	0	0	0	0	0	0	0	0	0	-1	10
0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: 0 1200 1700 1500 1400 1700 1400 1200 1900 1800 1600 1700 1100

EXAMPLE 4- 17 customers Groetschel (1992)
CAPACITY : 6

-1	121	518	142	84	297	35	29	36	236	390	238	301	55	96	153	336
0	-1	246	745	472	237	528	364	332	349	202	685	542	157	289	426	483
0	0	-1	268	420	53	239	199	123	207	165	383	240	140	448	202	57
0	0	0	-1	211	466	74	182	243	105	150	108	326	336	184	391	145
0	0	0	0	-1	70	567	191	27	346	83	47	68	189	439	287	254
0	0	0	0	0	-1	324	638	437	240	421	329	297	314	95	578	435
0	0	0	0	0	0	-1	353	282	110	324	61	208	292	250	352	154
0	0	0	0	0	0	0	-1	505	289	262	476	196	360	444	402	495
0	0	0	0	0	0	0	0	-1	259	555	372	175	338	264	232	249
0	0	0	0	0	0	0	0	0	-1	134	530	154	105	309	34	29
0	0	0	0	0	0	0	0	0	0	-1	80	572	196	77	351	63
0	0	0	0	0	0	0	0	0	0	0	-1	150	488	112	120	267
0	0	0	0	0	0	0	0	0	0	0	0	-1	412	227	169	383
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	91	661	228
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	257	390
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	633
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

EXAMPLE 5- 21 customers Groetschel (1992) /CAPACITY:7

-1	380	140	495	280	480	340	350	370	505	185	240	310	345	280	105	380	280	165	305	150
0	-1	240	290	590	140	480	255	205	220	515	150	100	170	390	425	255	395	205	220	155
0	0	-1	170	445	750	160	495	265	220	240	600	235	125	170	485	525	405	375	87	315
0	0	0	-1	450	270	625	345	660	430	420	440	690	77	310	380	180	215	190	545	225
0	0	0	0	-1	255	440	755	235	650	370	320	350	680	150	175	265	400	435	385	485
0	0	0	0	0	-1	265	480	420	235	125	125	200	165	230	475	310	205	715	650	475
0	0	0	0	0	0	-1	480	81	435	380	575	440	455	465	600	245	345	415	295	170
0	0	0	0	0	0	0	-1	655	235	585	555	750	615	625	645	775	285	515	585	190
0	0	0	0	0	0	0	0	-1	610	360	705	520	835	605	590	610	865	250	480	545
0	0	0	0	0	0	0	0	0	-1	68	440	575	27	320	91	48	67	430	300	90
0	0	0	0	0	0	0	0	0	0	-1	155	380	640	63	430	200	160	175	535	240
0	0	0	0	0	0	0	0	0	0	0	-1	370	320	700	280	590	365	350	370	625
0	0	0	0	0	0	0	0	0	0	0	0	-1	490	605	295	460	120	350	425	390
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	130	500	540	97	285	36	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	110	480	570	78	320	96
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	155	475	495	120	240
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	385	585	390	350
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	91	415	605
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	635	355
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	510
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

EXAMPLE 6-24 customers Groetschel (1992) /CAPACITY : 7

-1	121	142	99	84	35	29	42	36	220	70	126	55	249	104	178	60	96	175	153	146	47	135	169
0	-1	192	228	235	108	119	165	178	154	71	136	262	110	74	96	264	187	182	261	239	165	151	221
0	0	-1	250	99	89	221	105	189	160	147	349	76	138	184	235	138	114	212	39	40	46	136	96
0	0	0	-1	175	128	76	146	32	76	47	30	222	56	103	109	225	104	164	99	57	112	114	134
0	0	0	0	-1	261	43	200	232	98	200	171	131	166	90	227	195	137	69	82	223	90	176	90
0	0	0	0	0	-1	268	53	138	239	123	207	178	165	367	86	187	202	227	130	68	230	57	86
0	0	0	0	0	0	-1	290	139	98	261	144	176	164	136	389	116	147	224	275	178	154	190	79
0	0	0	0	0	0	0	-1	211	74	81	182	105	150	121	108	310	37	160	145	196	99	125	173
0	0	0	0	0	0	0	0	-1	54	219	92	82	119	31	43	58	238	147	84	53	267	170	255
0	0	0	0	0	0	0	0	0	-1	293	50	232	264	148	232	203	190	248	122	259	227	219	134
0	0	0	0	0	0	0	0	0	0	-1	219	83	172	149	79	139	134	112	126	62	199	153	97
0	0	0	0	0	0	0	0	0	0	0	-1	272	180	315	188	193	245	258	228	29	159	342	209
0	0	0	0	0	0	0	0	0	0	0	0	-1	70	191	121	27	83	47	64	68	173	119	148
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	214	223	49	185	123	115	86	90	313	151
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	185	86	124	156	40	124	95	82	207
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	243	209	286	159	190	216	229	225
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	134	154	63	105	34	29	22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	130	167	59	101	56	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	80	196	88	77	63
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	150	112	96	120
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	91	228	158
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	187	196
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	257
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

EXAMPLE 7-26 customers (http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/V/fri-n26-k3.vrp)/CAPACITY : 10

-1	181	197	161	190	182	190	160	148	128	121	103	99	107	130	130	95	51	51	81	79	37	27	58	107	90
0	-1	127	179	157	197	194	202	188	188	155	136	116	100	111	132	122	139	109	125	141	148	80	65	64	93
0	0	-1	220	268	241	278	272	280	257	250	223	210	190	178	189	212	205	196	154	157	186	186	128	102	51
0	0	0	-1	185	223	193	228	222	230	206	198	172	160	140	129	140	163	158	144	102	107	135	136	77	50
0	0	0	0	-1	157	180	147	180	173	181	156	148	122	111	92	83	93	116	113	94	53	64	87	90	26
0	0	0	0	0	-1	147	160	124	155	148	156	130	122	96	86	68	62	71	93	93	68	30	46	63	68
0	0	0	0	0	0	-1	185	165	125	139	128	135	98	78	74	82	77	87	87	100	109	39	38	29	13
0	0	0	0	0	0	0	-1	172	152	112	127	117	124	88	70	62	68	64	75	74	87	96	26	34	33
0	0	0	0	0	0	0	0	-1	181	175	135	156	146	153	119	103	91	91	80	85	89	106	112	54	22
0	0	0	0	0	0	0	0	0	-1	159	156	117	142	133	141	110	98	78	74	61	63	68	87	92	44
0	0	0	0	0	0	0	0	0	0	-1	152	127	86	102	93	100	66	54	37	43	42	56	53	62	73
0	0	0	0	0	0	0	0	0	0	0	-1	81	67	36	76	74	82	78	91	55	34	32	31	24	15
0	0	0	0	0	0	0	0	0	0	0	0	-1	95	68	31	66	62	71	63	76	40	20	27	34	23
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	99	89	54	89	84	92	77	83	47	26	11	11
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	98	98	64	100	95	103	88	92	56	36	18
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	110	95	58	88	82	90	71	75	39	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	114	84	44	70	62	71	52	59	22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	135	93	54	65	55	63	34	37
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	169	116	81	72	61	65	26
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	151	91	59	46	35	39
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	139	64	49	11	9
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	133	62	42	11
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	129	53	42
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	93	40
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	83
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

EXAMPLE 8-29 customers (http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/V/bayg-n29-k4.vrp) /CAPACITY : 8

-1	97	205	139	86	60	220	65	111	115	227	95	82	225	168	103	266	205	149	120	58	257	152	52	180	136	82	34	145
0	-1	129	103	71	105	258	154	112	65	204	150	87	176	137	142	204	148	148	49	41	211	226	116	197	89	153	124	74
0	0	-1	219	125	175	386	269	134	184	313	201	215	267	248	271	274	236	272	160	151	300	350	239	322	78	276	220	60
0	0	0	-1	167	182	180	162	208	39	102	227	60	86	34	96	129	69	58	60	120	119	192	114	110	192	136	173	173
0	0	0	0	-1	51	296	150	42	131	268	88	131	245	201	175	275	218	202	119	50	281	238	131	244	51	166	95	69
0	0	0	0	0	-1	279	114	56	150	278	46	133	266	214	162	302	242	203	146	67	300	205	111	238	98	139	52	120
0	0	0	0	0	0	-1	178	328	206	147	308	172	203	165	121	251	216	122	231	249	209	111	169	72	338	144	237	331
0	0	0	0	0	0	0	-1	169	151	227	133	104	242	182	84	290	230	146	165	121	270	91	48	158	200	39	64	210
0	0	0	0	0	0	0	0	-1	172	309	68	169	286	242	208	315	259	240	160	90	322	260	160	281	57	192	107	90
0	0	0	0	0	0	0	0	0	-1	140	195	51	117	72	104	153	93	88	25	85	152	200	104	139	154	134	149	135
0	0	0	0	0	0	0	0	0	0	-1	320	146	64	68	143	106	88	81	159	219	63	216	187	88	293	191	258	272
0	0	0	0	0	0	0	0	0	0	0	-1	174	311	258	196	347	288	243	192	113	345	222	144	274	124	165	71	153
0	0	0	0	0	0	0	0	0	0	0	0	-1	144	86	57	189	128	71	71	82	176	150	56	114	168	83	115	160
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	61	165	51	32	105	127	201	36	254	196	136	260	212	258	234
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	106	110	56	49	91	153	91	197	136	94	225	151	201	205
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	215	159	64	126	128	190	98	53	78	218	48	127	214
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	61	155	157	235	47	305	243	186	282	261	300	252
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	105	100	176	66	253	183	146	231	203	239	204
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	113	152	127	150	106	52	235	112	179	221
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	79	163	220	119	164	135	152	153	114
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	236	201	90	195	90	127	84	91
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	273	226	148	296	238	291	269
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	112	130	286	74	155	291
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	130	178	38	75	180
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	281	120	205	270
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	213	145	36
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	94	217
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	162
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Note: Due to the size of the next three examples, we will display them as a numbers not a matrix. In order to put these numbers in a format similar to the above examples, the following procedure must be applied.

If (a b c d e f) represent the cost then we can put them in the format as:

-1	a	b	c
0	-1	d	e
0	0	-1	f
0	0	0	-1

Where -1 assigned for the cost of traveling from a customer to himself and the cost below the diagonal is 0 and the given numbers organized above the diagonal.

EXAMPLE 9-31 customers Eilon, Watson-Gandy and Christofides (1971)
CAPACITY : 140

41 38 80 80 97 92 96 78 98 87 95 77 93 91 98 96 40 73 82 55 52 76 76 76 72 98 98
93 89 68 3 54 54 64 59 56 39 59 52 58 38 55 52 58 59 5 34 48 16 16 46 44 50 33 58
58 66 55 32 56 56 67 62 59 41 62 50 61 41 58 53 61 62 5 37 46 19 17 49 46 53 34
61 61 68 58 33 3 19 13 16 54 20 47 15 30 15 25 19 17 60 46 44 54 68 8 11 4 53 33
32 14 10 64 16 10 14 54 17 46 12 29 12 22 16 14 61 46 44 54 68 9 11 4 54 30 29 12
9 64 7 11 53 12 46 8 34 10 24 10 8 71 50 45 58 77 19 20 20 57 27 26 23 8 67 10 57
13 42 8 32 14 19 10 8 65 46 42 55 72 15 15 14 55 30 29 18 5 66 48 4 35 45 25 3
12 4 4 63 39 33 48 69 18 15 18 47 21 20 22 7 57 39 12 45 24 47 30 42 44 40 8 9
22 36 44 42 50 6 27 28 65 48 22 33 6 21 7 9 3 5 66 39 31 45 65 22 19 21 45 15
15 25 10 55 39 18 39 24 36 38 49 12 4 30 46 40 36 43 15 18 20 54 39 38 28 3 15 4
2 65 43 36 53 71 16 18 17 49 19 18 20 5 63 26 14 24 26 40 16 18 24 44 20 18 25 22
19 19 41 29 34 14 6 4 62 41 36 51 68 17 14 16 49 21 20 20 5 60 12 14 54 28 21 38
57 24 18 28 34 8 7 32 18 47 2 65 42 34 48 67 20 20 20 46 17 16 24 9 58 66 44 35
50 69 18 18 19 48 19 18 22 7 60 36 45 18 14 52 47 57 34 60 60 72 62 32 9 22 36 37
35 41 6 26 26 57 44 26 31 45 35 33 40 15 18 19 54 39 33 21 45 39 50 16 44 44 61
51 21 59 57 64 30 61 61 79 69 18 6 5 47 34 34 20 15 66 10 42 28 28 26 12 53 50 35
34 15 11 60 32 34 64 52 18 3 39 24 51 39 23 52 15 76 65

DEMAND: 0 24 34 11 15 11 1 3 29 6 25 6 25 2 28 8 10 18 45 33 17 9 16 35 5 60 80
39 95 90 123

EXAMPLE 10-42customers(http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/V/swiss-n42-k5.vrp)

CAPACITY : 9

0 15 30 23 32 55 33 37 92 114 92 110 96 90 74 76 82 67 72 78 82 159
122 131 206 112 57 28 43 70 65 66 37 103 84 125 129 72 126 141 183 124
15 0 34 23 27 40 19 32 93 117 88 100 87 75 63 67 71 69 62 63 96 164
132 131 212 106 44 33 51 77 75 72 52 118 99 132 132 67 139 148 186 122
30 34 0 11 18 57 36 65 62 84 64 89 76 93 95 100 104 98 57 88 99 130
100 101 179 86 51 4 18 43 45 95 45 115 93 152 159 100 112 114 153 94
23 23 11 0 11 48 26 54 70 94 69 89 75 84 84 89 92 89 54 78 99 141
111 109 190 89 44 11 29 54 56 89 47 118 96 147 151 90 122 126 163 101
32 27 18 11 0 40 20 58 67 92 61 78 65 76 83 89 91 95 43 72 110 141
116 105 190 81 34 19 35 57 63 97 58 129 107 156 158 92 129 127 161 95
55 40 57 48 40 0 23 55 96 123 78 75 62 36 56 66 63 95 37 34 137 174
156 129 224 90 15 59 75 96 103 105 91 158 139 164 156 78 169 163 191 115
33 19 36 26 20 23 0 45 85 111 75 82 69 60 63 70 71 85 44 52 115 161
136 122 210 91 25 37 54 78 81 90 68 136 116 150 147 76 148 147 180 111
37 32 65 54 58 55 45 0 124 149 118 126 113 80 42 42 49 40 87 60 94 195
158 163 242 135 65 63 79 106 101 50 66 118 104 109 103 36 160 178 218 153
92 93 62 70 67 96 85 124 0 28 29 68 63 122 148 155 156 159 67 129 148
78 80 39 129 46 82 65 55 40 61 157 97 159 135 212 221 159 110 72 95 35
114 117 84 94 92 123 111 149 28 0 54 91 88 150 174 181 182 181 95 157 159
50 65 27 102 65 110 87 73 50 68 176 112 166 142 229 241 184 99 46 69 38

92 88 64 69 61 78 75 118 29 54 0 39 34 99 134 142 141 157 44 110 161
103 109 52 154 22 63 68 66 61 81 158 107 175 151 216 219 150 137 100 115
37
110 100 89 89 78 75 82 126 68 91 39 0 14 80 129 139 135 167 39 98 187
136 148 81 186 28 61 92 97 98 117 173 134 204 181 232 229 153 176 137 143
62
96 87 76 75 65 62 69 113 63 88 34 14 0 72 117 128 124 153 26 88 174
136 142 82 187 32 48 79 85 89 106 159 121 191 168 219 216 140 168 134 145
64
90 75 93 84 76 36 60 80 122 150 99 80 72 0 59 71 63 116 56 25 170 201
189 151 252 104 44 95 111 130 138 130 127 192 174 186 172 90 205 193 214 135
74 63 95 84 83 56 63 42 148 174 134 129 117 59 0 11 8 63 93 35 135 223
195 184 273 146 71 95 113 138 138 81 107 159 146 132 113 32 200 209 243 171
76 67 100 89 89 66 70 42 155 181 142 139 128 71 11 0 11 54 103 46 130
230 198 192 279 155 80 99 117 143 141 74 107 155 143 122 102 22 202 215 250
179
82 71 104 92 91 63 71 49 156 182 141 135 124 63 8 11 0 65 100 39 140
232 203 192 281 153 78 103 121 147 146 85 115 164 152 133 112 33 208 218 251
178
67 69 98 89 95 95 85 40 159 181 157 167 153 116 63 54 65 0 127 92 83
224 180 199 269 175 106 95 109 135 125 21 80 107 100 71 63 33 173 205 249
191
72 62 57 54 43 37 44 87 67 95 44 39 26 56 93 103 100 127 0 67 153 145
139 96 196 53 23 60 70 81 95 134 101 172 149 194 190 115 160 138 159 80
78 63 88 78 72 34 52 60 129 157 110 98 88 25 35 46 39 92 67 0 152 207
188 162 258 119 48 89 107 129 134 108 114 176 159 163 147 66 200 197 224 147
82 96 99 99 110 137 115 94 148 159 161 187 174 170 135 130 140 83 153 152
0 188 128 184 222 183 139 95 95 110 91 62 54 24 23 81 110 113 108 164 217
184
159 164 130 141 141 174 161 195 78 50 103 136 136 201 223 230 232 224 145
207 188 0 65 57 51 109 160 132 116 90 102 217 148 188 168 264 281 231 100
26 30 75
122 132 100 111 116 156 136 158 80 65 109 148 142 189 195 198 203 180 139
188 128 65 0 91 94 126 145 100 82 60 57 167 99 126 106 208 230 194 36 39
94 103
131 131 101 109 105 129 122 163 39 27 52 81 82 151 184 192 192 199 96 162
184 57 91 0 106 53 115 104 94 74 94 196 134 192 168 251 260 197 126 64 64
19
206 212 179 190 190 224 210 242 129 102 154 186 187 252 273 279 281 269 196
258 222 51 94 106 0 158 211 180 163 136 145 259 190 218 200 302 323 278 120
65 49 124
112 106 86 89 81 90 91 135 46 65 22 28 32 104 146 155 153 175 53 119 183
109 126 53 158 0 75 89 88 83 103 178 129 197 173 236 238 166 156 111 115
34
57 44 51 44 34 15 25 65 82 110 63 61 48 44 71 80 78 106 23 48 139 160
145 115 211 75 0 53 68 86 95 114 90 160 139 173 168 92 162 150 176 101
28 33 4 11 19 59 37 63 65 87 68 92 79 95 95 99 103 95 60 89 95 132
100 104 180 89 53 0 18 44 45 92 42 112 89 149 156 99 111 116 155 97
43 51 18 29 35 75 54 79 55 73 66 97 85 111 113 117 121 109 70 107 95
116 82 94 163 88 68 18 0 27 27 103 42 109 85 157 168 115 94 98 140 90

70 77 43 54 57 96 78 106 40 50 61 98 89 130 138 143 147 135 81 129 110
90 60 74 136 83 86 44 27 0 21 128 62 119 96 179 192 142 79 72 115 74
65 75 45 56 63 103 81 101 61 68 81 117 106 138 138 141 146 125 95 134 91
102 57 94 145 103 95 45 27 21 0 115 46 98 75 163 179 136 67 81 129 95
66 72 95 89 97 105 90 50 157 176 158 173 159 130 81 74 85 21 134 108 62
217 167 196 259 178 114 92 103 128 115 0 69 86 81 60 65 54 158 195 243 190
37 52 45 47 58 91 68 66 97 112 107 134 121 127 107 107 115 80 101 114 54
148 99 134 190 129 90 42 42 62 46 69 0 71 49 117 133 98 95 127 175 132
103 118 115 118 129 158 136 118 159 166 175 204 191 192 159 155 164 107 172
176 24 188 126 192 218 197 160 112 109 119 98 86 71 0 24 94 127 137 100
163 218 194
84 99 93 96 107 139 116 104 135 142 151 181 168 174 146 143 152 100 149 159
23 168 106 168 200 173 139 89 85 96 75 81 49 24 0 104 133 127 85 143 197
170
125 132 152 147 156 164 150 109 212 229 216 232 219 186 132 122 133 71 194
163 81 264 208 251 302 236 173 149 157 179 163 60 117 94 104 0 39 100 190
241 292 246
129 132 159 151 158 156 147 103 221 241 219 229 216 172 113 102 112 63 190
147 110 281 230 260 323 238 168 156 168 192 179 65 133 127 133 39 0 81 216
259 307 253
72 67 100 90 92 78 76 36 159 184 150 153 140 90 32 22 33 33 115 66 113
231 194 197 278 166 92 99 115 142 136 54 98 137 127 100 81 0 193 214 253
187
126 139 112 122 129 169 148 160 110 99 137 176 168 205 200 202 208 173 160
200 108 100 36 126 120 156 162 111 94 79 67 158 95 100 85 190 216 193 0 74
129 137
141 148 114 126 127 163 147 178 72 46 100 137 134 193 209 215 218 205 138
197 164 26 39 64 65 111 150 116 98 72 81 195 127 163 143 241 259 214 74 0
55 80
183 186 153 163 161 191 180 218 95 69 115 143 145 214 243 250 251 249 159
224 217 30 94 64 49 115 176 155 140 115 129 243 175 218 197 292 307 253 129
55 0 81
124 122 94 101 95 115 111 153 35 38 37 62 64 135 171 179 178 191 80 147
184 75 103 19 124 34 101 97 90 74 95 190 132 194 170 246 253 187 137 80 81
0

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

EXAMPLE 11-48 customers Held and Karp (1970)
CAPACITY: 15

0 273 0 1272 999 0 744 809 1519 0 1138 866 140 1425
0 1972 1722 937 1861 1052 0 1580 1338 697 1473 776 400 0
1878 1640 951 1713 1049 182 304 0 1539 1226 267 1761 402 820
699 884 0 1457 1185 227 1617 361 721 538 755 177 0 429
440 1229 370 1119 1735 1335 1612 1486 1362 0 1129 894 587 1073
578 851 454 749 757 587 891 0 1251 992 369 1304 406 740
393 690 506 335 1082 252 0 1421 1173 554 1369 618 551 173
476 609 435 1199 308 222 0 588 334 721 1092 581 1551 1198

1501 981 930 726 803 814 1025 0 334 358 1212 453 1095 1769
1370 1654 1474 1358 96 920 1094 1227 663 0 837 626 739 798
670 1159 760 1049 967 819 583 309 510 617 632 610 0 1364
1124 596 1283 641 613 216 516 681 504 1125 238 235 90 999
1156 546 0 229 358 1291 973 1152 2072 1692 1995 1552 1496 653
1252 1335 1525 572 557 983 1479 0 961 847 1114 565 1060 1300
919 1149 1317 1153 563 569 820 835 972 642 397 745 1163 0
754 533 701 1315 567 1605 1286 1580 936 927 947 940 892 1114
225 879 821 1105 676 1183 0 1169 915 426 1204 443 807 435
739 594 428 986 165 100 263 763 1000 411 240 1264 725 865
0 1488 1219 285 1796 374 1017 879 1079 197 341 1493 863 626
770 908 1467 1023 831 1473 1399 821 699 0 720 481 676 846
579 1251 861 1161 928 803 560 414 541 700 451 558 180 645
839 549 644 453 950 0 1280 1009 155 1447 235 818 548 815
316 180 1183 454 219 400 767 1178 651 442 1326 1004 790 290
410 624 0 816 543 456 1143 325 1259 913 1214 723 649 813
552 524 740 293 780 478 723 847 869 388 483 690 325 479
0 664 937 1936 959 1802 2596 2198 2485 2203 2119 882 1745 1897
2049 1240 831 1438 1983 801 1427 1374 1809 2147 1356 1941 1480 0
1178 915 319 1275 331 826 483 780 500 343 1033 269 90 311
726 1038 476 316 1254 818 803 107 594 480 188 435 1829 0
939 667 337 1213 217 1137 803 1100 604 521 902 482 410 630
420 879 485 623 976 882 484 384 590 369 350 129 1603 320
0 1698 1441 604 2085 665 1255 1181 1347 482 652 1763 1188 952
1087 1111 1726 1333 1152 1643 1716 968 1024 326 1241 736 949 2339
919 872 0 983 812 907 742 862 1123 731 985 1104 939 642
355 805 630 862 700 235 543 1157 214 1056 511 1191 413 792
708 1524 605 699 1511 0 1119 848 214 1309 182 943 627 916
455 340 1032 397 238 459 617 1023 525 470 1169 902 655 251
499 473 161 325 1780 154 197 815 697 0 1029 776 424 1479
312 1359 1086 1361 630 649 1131 833 706 924 443 1082 827 939
983 1222 318 712 504 680 547 355 1673 623 358 669 1051 469
0 1815 1560 748 1760 864 188 292 260 641 533 1604 713 570
405 1374 1631 1022 482 1905 1210 1420 646 838 1097 632 1081 2421
652 957 1092 1018 761 1171 0 721 526 817 703 732 1282 883
1171 1058 918 463 432 622 739 586 488 123 669 878 390 794
525 1098 166 745 492 1315 580 529 1397 290 607 847 1144 0
1753 1494 666 1727 783 271 279 328 562 451 1556 666 503 360
1299 1579 973 443 1836 1184 1341 585 758 1038 552 1007 2394 582
881 1019 985 685 1089 83 1094 0 330 598 1592 872 1456 2300
1906 2202 1857 1783 663 1453 1581 1749 887 586 1155 1690 346 1225
1017 1499 1794 1049 1607 1137 357 1508 1263 1982 1280 1446 1316 2145
1036 2083 0 1499 1244 521 1479 608 483 178 445 528 362 1298
410 257 115 1070 1320 715 205 1590 949 1137 330 703 781 375
779 2136 344 660 1010 743 472 919 317 836 259 1828 0 1107
1304 2172 686 2066 2540 2156 2385 2425 2290 947 1758 1985 2055 1633
982 1475 1969 1286 1239 1836 1885 2439 1497 2115 1759 825 1950 1849
2708 1427 1969 2063 2445 1371 2412 1005 2165 0 1576 1306 356 1698
491 609 490 665 220 130 1461 642 396 428 1057 1463 902 510
1621 1210 1056 495 414 905 296 774 2237 429 645 695 996 457

776 426 1008 345 1903 330 2377 0 942 685 467 1057 400 1038
662 966 704 568 795 262 309 492 547 796 273 455 1034 660
679 231 751 238 392 291 1589 242 240 1061 466 254 598 875
354 811 1272 559 1723 667 0 484 668 1583 387 1466 2099 1699
1969 1845 1727 371 1260 1453 1568 999 371 953 1492 689 863 1200
1356 1837 925 1547 1148 579 1402 1250 2089 987 1393 1434 1972 833
1925 504 1668 636 1829 1162 0 617 444 882 1252 744 1776 1430
1729 1122 1105 882 1051 1039 1256 252 802 882 1238 503 1207 189
999 1011 702 959 516 1204 949 631 1148 1110 823 507 1584 828
1507 849 1291 1720 1235 792 1087 0 896 1157 2139 904 2013 2699
2300 2568 2405 2301 967 1858 2043 2166 1483 940 1550 2091 995 1446
1645 1949 2374 1506 2121 1688 347 1986 1802 2594 1584 1963 1926 2571
1429 2523 653 2264 534 2410 1744 600 1490 0 1184 1359 2182 668
2082 2493 2117 2333 2428 2285 973 1737 1972 2026 1681 1021 1467 1938
1376 1197 1891 1872 2455 1506 2114 1785 959 1943 1867 2734 1395 1975
2101 2408 1369 2380 1114 2138 145 2367 1724 701 1787 678 0 1030
1176 1961 443 1865 2266 1888 2108 2204 2059 768 1508 1744 1796 1489
826 1240 1709 1239 969 1704 1644 2237 1287 1890 1573 940 1717 1650
2520 1166 1752 1898 2179 1146 2151 1019 1908 290 2139 1500 550 1614
727 229 0 1718 1475 781 1600 875 264 138 177 738 595 1472
592 514 303 1326 1508 898 354 1828 1042 1403 567 928 998 641
1038 2336 603 923 1212 861 739 1187 194 1021 220 2044 268 2281
519 796 1835 1553 2435 2238 2010 0 604 335 678 930 552 1398
1023 1327 945 853 588 598 661 853 236 550 396 813 674 741
442 591 921 216 676 231 1266 582 341 1176 626 515 548 1231
352 1163 932 917 1531 972 361 917 486 1461 1560 1353 1157 0

DEMAND: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Bibliography

Achuthan, N.R. and Caccetta, L. (1991). Integer Linear Programming Formulation for a Vehicle Routing Problem. *European Journal of Operational Research,* **52** pp 86-89.

Achuthan, N.R., Caccetta, L. and Hill, S. P. (1996). A New Subtour Elimination Constraint for the Vehicle Routing Problem. *European Journal of Operational Research,* **91** pp. 573-586.

Achuthan, N.R., Caccetta, L. and Hill, S. P. (2003). An Improved Branch and Cut Algorithm for the Capacitated Vehicle Routing Problem *Transportation Science,* **37** pp. 153-169

Agarwal, Y., Mathur, K. and Salkin, H.M. (1989). A Set Partitioning Based Exact Algorithm for the Vehicle Routing Problem. *Networks,* **19** pp 731-749.

A Gomory cut (1998).A cut, Michael A. Trick. Viewed 11/07/2009, <http://mat.gsia.cmu.edu/orclass/integer/node15.html>.

AllBusiness.com, Inc. (2007), Council of Logistics Management San Francisco U.S.A. Viewed 31/12/2008.< http://www.allbusiness.com/company-activities-management/operations-supply/8898081-1.html>.

Altinkemer, K. and Gavish, B. (1991). Parallel Saving Based Heuristics for the Delivery Problem. *Operations Research,* **39** pp 456-469.

Amberg,A., Domschke, W. and Voss, S. (2000). Multiple Center Capacitated Arc Routing Problems: A Tabu Search Algorithm using Capacitated Trees. *European Journal of Operational Research,* **124** pp 360-376.

Araque, J. R. (1989). Contributions to the Polyhedral Approach to the Vehicle Routing Problem. *Ph.D Dissertation, State University of New York at Stony Brook.*

Araque, J. R., Kudva, G., Morin, T. L. and Pekny, J. F. (1994). A Branch-and-Cut Algorithm for Vehicle Routing Problems. *Annals of Operations*

Research, **50** pp 37 - 59.

Archetti,C. Hertz,A. Speranza,M.G. (2006a). A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem, *Transportation Science,* **40** pp 64-73.

Archetti, C. Savelsbergh, M. Speranza, M.G. (2006 b). Worst-Case Analysis for Split Delivery Vehicle Routing Problems, *Transportation Science* **40** pp 226-234.

Atkinson, J. B. (1994). A Greedy Look-ahead Heuristic for Combinatorial Optimization: An Application to Vehicle Scheduling with Time Windows. *Journal of the Operational Research Society***, 45** pp 673 - 684.

Attanasio, A. Cordeau J. Ghiani, G. and Laporte, G. (2003). Parallel Tabu Search Heuristics for the Dynamic Multi-Vehicle Dial-a-Ride Problem. Working paper, University of Calabria, Italy.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D. and Rinaldi G. (1995). Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem. *Research Report 949-M, Universite Joseph Fourier, Grenoble, France.*

Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y. and Taillard, E.(1997). A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research,* **5** pp109-122.

Balas, E. and Toth, P. (1985). Branch and Bound Methods, in *The Traveling Salesman Problem* (Lawler E.L., Lenstra J.K., Rinnooy Kan A. G. H. and Shmoys D. B., Editors) John Wiley and Sons, pp 361 - 401.

Baldacci, R. and Mingozzi, A. (2006). Lower Bounds and an Exact Method for the Capacitated Vehicle Routing Problem**.** *International Conference on Service Systems and Service Management,* **2** pp 1536 – 1540.

Balinski, M. and Quandt, R. (1964). On an Integer Program for a Delivery Problem. *Operations Research,* **12** pp 300 - 304.

Bent, R.W. and Van Hentenryck, P. (2004). Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. *Operations research,* **52(6)** pp 977-987.

Berger, J. and Barkaoui, M. (2004), A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computer & OR*. **12**, pp 2037-2053.

Bodin, L. and Golden, G. (1981). Classification in Vehicle Routing and Scheduling. *Networks,* **11** pp 97-108.

Bodin, L. D., Golden, B. L., Assad, A. A. and Ball, M. 0. (1983). Routing and Scheduling of Vehicles and Crews, the State of the Art. *Computers and Operations Research,* **10** pp 69-211.

Boyd, E. A. (1994). Fenchel Cutting Planes for Integer Programs. *Operations Research,* **42** pp 53 - 64.

Brodie, G. R. and Waters, C. D. J. (1998). Integer Linear Programming Formulation for Vehicle Routing Problems. *European Journal of Operational Research,* **34** pp 403 - 404.

Campos, V., Corberan, A. and Mota, E. (1991). Polyhedral Results for a Vehicle Routing Problem. *European Journal of Operational Research,* **52** pp 75 - 85.

Carpaneto, G., Dell'Amico, M., Fischetti, M. and Toth, P. (1989). A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Networks,* **19** pp 531 - 548.

Chao, I.M., Golden, B., Wasil, E., (1995). An improved heuristic for the period vehicle routing problem. *Networks*. **26(1)** pp 25—44.

Caseau, Y. Silverstein, G. Laburthe, F. (2001). Learning Hybrid Algorithms for Vehicle Routing Problems. *Theory and Practice of Logic Programming*. **1** pp 779-806.

Christofides, N. (1976). The Vehicle Routing Problem. *RAIRO, recherche operationnelle,* **10** pp 55 - 70.

Christofides, N. (1979). The Travelling Salesman Problem, in *Combinatorial*

Optimization (Christofides, N., Mingozzi, A., Toth, P. and Sandi, C, Editors) Wiley, pp 131 - 149.

Christofides, N. (1985). Vehicle Routing, in *The Traveling Salesman Problem* (Lawler E.L., Lenstra J.K., Rinnooy Kan A. G. H. and Shmoys D. B., Editors) John Wiley and Sons, pp 431 - 448.

Christofides, N. and Eilon, S. (1969). An Algorithm for the Vehicle Dispatching Problem. *Operations Research,* **20** pp 309 - 318.

Christofides, N., Mingozzi, A. and Toth, P. (1981a). Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations. *Mathematical Programming,* **20** pp 255-282.

Christofides, N., Mingozzi, A. and Toth, P. (1981b). State Space Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks,* **11** pp 145-164.

Christofides, N., Mingozzi, A. and Toth, P. (1979). The Vehicle Routing Problem, in *Combinatorial Optimization* (Christofides, N., Mingozzi, A.,Toth, P. and Sandi, C, Editors) Wiley, Chichester, pp 315 - 338.

Clarke, G. and Wright, J. (1964). Scheduling of Vehicles From a Central Depot to a Number of Delivery Points. *Operations Research,* **12** pp 568 - 581.

Computational Infrastructure for Operations Research 2003, Vehicle Routing Data Sets USA, viewed 31 October 2006, http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data.

Cordeau, J.F., Gendreau, M., Laporte, G., (1997). A Tabu Search heuristic for periodic and multi-depot vehicle routing problems. *Networks,* **30(2)** pp 105—119.

Cordeau, J.-F., Laporte, G. and Mercier, A.(2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, **52** pp 928-936.

Cornuejols, G. (2007). Revival of the Gomory Cuts in the 1990s. *Annals of Operations Research*, **149** pp. 63-66.

Cornuéjols, G. and Harche, F. (1993). Polyhedral Study of the Capacitated Vehicle Routing Problem. *Mathematical Programming,* **60** pp 21 - 52.

Dantzig, G. B., Fulkerson, D. R. and Johnson, S. M. (1954). Solution of a Large Scale Travelling Salesman Problem. *Operations Research,* **2** pp 393 - 410.

Dantzig, G. B., Fulkerson, D. R. and Johnson, S. M. (1959a). On a Linear Programming, Combinatorial Approach to the Traveling-Salesman Problem. *Operations Research,* **7** pp 58 - 66.

Dantzig, G. B. and Ramser, J. H. (1959b). The Truck Dispatching Problem. *Management Science,* **6** pp 80 - 91.

Desrochers, M., Desrosiers, J. and Solomon, M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research,* **40** pp 342 - 354.

Desrochers, M. and Laporte, G. (1991). Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints. *Operations Research Letters,* **10** pp 27-36.

Desrochers M., Lenstra J. K. and Savelsbergh M. W. P. (1990) A Classi-fication Scheme for Vehicle Routing and Scheduling Problems. *European Journal of Operational Research,* **46** pp 322-332.

Dueck, G.(1993). New Optimization Heuristics: The Great Deluge Algorithm the Record-To-Record Travel. *Journal of Computational Physics,* **104** pp 86-92.

Dueck, G. and Scheurer,T.(1990). Threshold Accepting: A General Purpose Optimization Algorithm. *Journal of Computational Physics,* **90** pp 161-175.

Eilon, S., Watson-Gandy, C. and Christofides, N. (1971). Distribution Management, Mathematical Modelling and Practical Analysis. *Griffin,* London.

Erera, A. and Daganzo, C. (2003). A Dynamic Scheme for Stochastic Vehicle Routing. Working paper. Georgia Institute of Technology, U.S.A

Ferland, J.A. and Mechelon, P. (1988). The Vehicle Scheduling Problem with Multiple Vehicle Types. *Journal of the Operational Research Society,* **39(6)** pp 577-583.

Fischetti M., Toth P. and Vigo D. (1994). A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs. *Operations Research,* **42** pp 846 - 859.

Fisher, M. (1994a). Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees. *Operations Research,* **42** pp 626 - 642.

Fisher, M. and Jaikumar, R. (1981). A Generalised Assignment Heuristic for Vehicle Routing. *Networks,* **11** pp 109-124.

Forbes, M.A., Holt, J.N. and Watts, A.M. (1994). An Exact Algorithm for Multiple Depot Bus Scheduling. *European Journal of Operational Research,* **72** pp 115-124.

Foster, B. A. and Ryan, D. M. (1976). An Integer Programming Approach to the Vehicle Scheduling Problem. *Operational Research Quarterly,* **27** pp 367 - 384.

Fraser, A. (1957).Simulation of Genetic Systems by Automatic Digital Computers. *Australian Journal of Biological Sciences*, **10** pp 484-491.

Garvin, W. M., Crandall, H. W., John, J. B. and Spellman, R. A. (1957). Applications of Linear Programming in the Oil Industry. *Management Science,* **3** pp 407 - 430.

Gaskell, T. J. (1967).Bases for Vehicle Fleet Scheduling. *Operational Research Quarterly,* **18** pp 281 - 295.

Gendreau, M., Laporte, G., Musaraganyi, C., and Taillard, E.D. (1999). A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Computers & Operations Research.* **26(12)** pp 1153-1173.

Giosa, D., Tansini, L. and Viera, O. (2002). New Assignment Algorithms for the Multi-Depot Vehicle Routing Problem. *Journal of the Operational Research Society,* **53** pp 977-984

Goetschalckx, M. and Jacobs-Blecha, C. (1989). The Vehicle Routing Problem with Backhaul. *European Journal of Operational Research,* **42** pp 39-51.

Goldberg, D.E (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley. NY

Golden, B. L. and Assad, A. A. (1986). Perspectives on Vehicle Routing: Exciting News Developments. *Operations Research,* **34** pp 803 - 810.

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

Groetschel, M., Monma, C.L AND Stoer, M. (1992). Computational results with a cutting plane algorithm for designing communication networks with low connectivity constraints. *Operations Research,* **40** pp 309-330.

Hadjiconstantinou, E., Christofides, N. and Mingozzi, A. (1995). A New Exact Algorithm for the Vehicle Routing Problem Based on *q*-paths and *k*-Shortest Paths Relaxations. *Annals of Operations Research,* **61** pp 21 - 44.

Hall, R. W., Du, Y. and Lin, J. (1994). Use of Continuous Approximations within Discrete Algorithms for Routing Vehicles: Experimental Results and Interpretation. *Networks,* **24** pp 43 - 56.

Held, M., and Karp, R. M. (1970). The Traveling Salesman Problem and Minimum Spanning Trees. *Operations Research*, **18** pp 1138-1162.

Held, M., and Karp, R. M. (1971). The Traveling Salesman Problem and Minimum Spanning Trees: part II. *Mathematical Programming,* **1** pp 6-25.

Hernandez-Perez, H. and Salazar-Gonzalez, J. (2004).A Branch-and-Cut Algorithm for A Traveling Salesman Problem with Pickup and Delivery, *Discrete Applied Mathematics*, **145** pp 126–139.

Hooker, J. N. (2002).Logic, optimization and constraint programming.

INFORMS Journal on Computing, **14** pp 295-321.

Hooker, J. N. (2005). A hybrid method for planning and scheduling, *Constraints* **10** pp 385-401.

Hooker, J. N. (2006). An integrated method for planning and scheduling to minimize tardiness. *Constraints,* **11** pp 139-157.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research,* **55** pp 588-602.

ILOG S.A. *ILOG Dispatcher 4.0 User's Manual*. ILOG S.A., 9 Rue de Verdun, 94253 Gentilly Cedex, France.

ILOG S.A. *ILOG Solver 6.0 User's Manual*. ILOG S.A., 9 Rue de Verdun, 94253 Gentilly Cedex, France.

Khoury, B. and Pardalos, P. (1995). (LNCS),An exact branch and bound algorithm for the Steiner Problem in Graphs. SpringerLink. **959** pp 582-590.

Kilby,P. Prosser,P. and Shaw,P. (1998). Guided local search for the vehicle routing problem. In *Proceedings of the 2nd International Conference on Metaheuristics*. INRIA, France, 1998

Kim, N.M. Rim, S.C. and Min, B.D. (1997). A Heuristic Algorithm for Vehicle Routing Problem with Backhauls. *International Journal of Management science,* **3(1)** pp 1-14.

Kleywegt, A.J. Nori, V.S. and Savelsbergh, M.P. (2002). Dynamic Programming Approximations for a Stochastic Inventory Routing Problem. Working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, USA.

Kolen, A. W. J., Rinnooy Kan, H. G. and Trienekens, H. W. J. M. (1987). Vehicle Routing with Time Windows. *Operations Research,* **35** pp 266 - 273.

Kulkarni, R. V. and Bhave, P. R. (1985). Integer Programming Formulations of Vehicle Routing Problems. *European Journal of Operational Research,* **20** pp 58 - 67.

Land, A.H. and Doig, A.G. (1960). An automatic method for solving discrete programming problems. *Econometrica* **28** pp 497-520.

Land, A. H. and Powell, S. (1973). FORTRAN Codes for Mathematical Programming: Linear, Quadratic and Discrete. Wiley, New York.

Langevin, A., Soumis, F. and Desrosiers, J. (1990). Classification of Traveling Salesman Problem Formulations. *Operations Research Letters,* **9** pp 127 - 132.

Laporte, G. (1986). Generalized Subtour Elimination Constraints and Connectivity Constraints. *Journal of the Operational Research Society,* **37** pp 509-514.

Laporte, G. (1992a). The Travelling Salesman Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research,* **59** pp 231-247.

Laporte, G. (1992). The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research,* **59** pp 345-358.

Laporte, G. and Bourjolly, J. M. (1984). Some Further Results on K-Star Constraints and Comb Inequalities. *Cahiers Du GERAD, G-82-10, Ecole des Hautes Etudes Commerciales de Montréal.*

Laporte, G., Desrochers, M. and Nobert, Y. (1984). Two Exact Algorithms for the Distance Constrained Vehicle Routing Problem. *Networks,* **14** pp 161 - 172.

Laporte,G. Louveaux, F.V. and Van Hamme, L. (2002). An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. *Operations research,* **50(3)** pp 415-423.

Laporte, G., Mercure, H. and Nobert, Y. (1986). An Exact Algorithm for the Asymmetrical Capacitated Vehicle Routing Problem. *Networks,* **16** pp 33 - 46.

Laporte, G., Mercure, H. and Nobert, Y. (1992). A Branch and Bound Algorithm for a Class of Asymmetrical Vehicle Routing Problems. *Journal* *of the Operational Research Society,* **43** pp 469-481.

Laporte, G. and Nobert Y. (1980). A Cutting Plane Algorithm for the m-Salesman Problem. *Journal of the Operational Research Society,* **31** pp 1017 - 1023.

Laporte, G. and Nobert Y. (1983). A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem. *Operations Research Spectrum,* **5** pp 77 - 85.

Laporte, G. and Nobert Y. (1984). Comb Inequalities for the Vehicle Routing Problem. *Methods of Operational Research,* **51** pp 271 - 276.

Laporte, G. and Nobert, Y. (1987). Exact Algorithms for the Vehicle Routing Problem. *Annals of Discrete Mathematics,* **31** pp 147-184.

Laporte, G., Nobert, Y. and Desrochers, M. (1985). Optimal Routing Under Capacity and Distance Restrictions. *Operations Research,* **33** pp 1050-1073.

Lenstra, J. K. and Rinnooy Kan, A. H. G. (1981). Complexity of Vehicle Routing and Scheduling Problems. *Networks,* **11** pp 221-227.

Li, C, Simchi-Levi, D. and Desrochers, M. (1991). On the Distance Constrained Vehicle Routing Problem. *Operations Research,* **40** pp 790-799.

Li, F., Golden, B. and Wasil, E. (2007). A Record-to-Record Travel Algorithm for Solving the Heterogeneous Fleet Vehicle Routing Problem. *Computers & Operations Research,* **34** pp 2734-2742.

Li, F., Golden, B. and Wasil, E. (2005). Very large-scale vehicle routing: New test problems, algorithms, and results. *Computers & Operations Research,* **32(5)** pp 1165–1179.

Magnanti, T. L. (1981). Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects. *Networks,* **11** pp 179 - 213.

Malik, K. and Yu, G. (1993). A Branch and Bound Algorithm for the Capacitated Minimum Spanning Tree Problem. *Networks,* **23** pp 525 - 532.

McCune, W. (2003). Mace4 reference manual and guide. Technical Report ANL/MCSTM-264, Mathematics and Computer Science Division, Argonne National Laboratory,Argonne.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953). Equation of state calculation by fast computing machines, *Journal of Chemistry and Physics.* **21** pp 1087-1091

Min, H. (1989). The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pick-up Points, *Transportation Research Part A,***23(5)** pp 377–386.

Montanari,U. (1974). Networks of constraints: Fundamental properties and application to picture processing. Information Science, 7, Also Technical Report, Carnegie Mellon University, 1970.

Naddef, D. (1994). A Remark on "Integer Linear Programming Formulation for a Vehicle Routing Problem" by N.R. Achuthan and L. Caccetta, or How to Use the Clark and Wright Savings to Write Such Integer Linear Programming Formulations. *European Journal of Operational Research,* **75** pp 238-241.

Nemhauser, G. L. and Wolsey, L. A. (1988). Integer and Combinatorial Optimization. *John Wiley and Sons.*

Nelson, M. D., Nygard, K. E., Griffin, J. H. and Shreve, W. E. (1985). Implementation Techniques for the Vehicle Routing Problem. *Computers and Operations Research,* **12** pp 273 - 283.

Padberg, M. and Rinaldi, G. (1987). Optimization of a 532-Symmetric Travelling Salesman Problem *Operations Research Letters,* **6** pp 1 - 7.

Paessens, H. (1988). The Savings Algorithm for the Vehicle Routing Problem. *European Journal of Operational Research,* **34** pp 336-344.

Prosser, P. and Shaw, P. (1996) . Study of Greedy Search with Multiple Improvement Heuristics for Vehicle Routing Problems. Working Paper, University of Strathclyde, Glasgow, Scotland.

Rayward-Smith, V.J. Osman, I.H. Reeves, C.R. and Smith, G.D. (1996) Modern Heuristic Search Methods, *John Wiley & Sons*.

Reimann, M., Doerner, K. and Hartl, R.F.(2004). D-ants: savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, **31** pp 563-591.

Reinelt. G. (1981). A Travelling Salesman Problem Library. *ORSA Journal of Computing,* **3** pp 376 - 384.

Ribeiro. C. C. and Soumis, F. (1994). A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. *Operations Research,* **42** pp 41 - 52.

Roman Barták (1998). Department of Theoretical Computer Science and Mathematical Logic. Charles University in Prague. Viewed 11/07/2009. http://ktiml.mff.cuni.cz/~bartak/constraints/intro.html .

Rousseau, L. Gendreau M. and Pesant, G. (1999). Using Constraint-based Operators with Variable Neighborhood Search to Solve the Vehicle Routing Problem with Time Windows. Presented at the CP-AI-OR'99 Workshop, February 25.-26., University of Ferrara, Italy

Salhi, S. and Nagy, G. (1999). A cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauls. *Journal of the Operation Research Society,* **50** pp 1034-1042.

Salhi, S. and Rand, G. (1993). Incorporating Vehicle Routing into the Vehicle Fleet Composition Problem. *European Journal of Operational Research,* **66(3)** pp 313-330.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research,* **35** pp 254 - 265.

Stewart, W. and Golden, B. (1983). Stochastic Vehicle Routing: A Comprehensive Approach. *European Journal of Operational Research,* **14** pp 371-385.

Taillard, E. (1993a). A Heuristic Column Generation Method for the Heterogeneous Fleet VRP. *RAIRO Recherche Operationnelle,* **33(1)** pp 1-14.

Taillard, E. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks,* **23** pp 661 - 673.

Tang Montane, F.A. and R. D. Galv˜ao,R.D.(2006) A Tabu Search Algorithm for the Vehicle Routing Problem with Simultaneous Pick-up and Delivery Service, *Computers & Operations Research*, **33(3)** pp. 595–619.

Thompson, P. M. and Psaraftis, H. N. (1993). Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling Problems. *Operations Research,* **41** pp 935 - 946.

Tian, Y., Song, J., Yao, D. and Hu, J. (2003). Dynamic vehicle routing problem using hybrid ant system, *The Proceedings of the 2003 IEEE Intemational Confuence on Intelligent Transportation Systems*, **1(2)** pp 970- 974.

Toth,P. Vigo, D.(2002). The Vehicle routing problem.*SIAM Monographs on discrete mathematics and application. NY*.

Van breedam, A. (1994). An analysis of the behavior of heuristics for vehicle routing problem for a selection of problems with vehicle related, customer related, and time related constraints .Ph.D. dissertation. University of Antwerp.

Vianna, D.S. Ochi, L.S. and Drummond, L.M. (1999). A Parallel Hybrid Evolutionary Metaheuristic for the Period Vehicle Routing Problem. In Proceedings of the 1999 Workshop on Biologically Inspired Solutions to Parallel Processing Problems.

Waltz, D. (1972). Generating Semantic Descriptions from Drawings of Scenes with Shadows. Ph. D. Thesis, MIT.

Waters, C. D. J (1988). Expanding the Scope of Linear Programming Solutions for Vehicle Scheduling Problems. *OMEGA International Journal of Management Science,* **16** pp 577 - 583.

Wren, A., and Holliday, A. (1972), Computer scheduling of vehicles from one or more depots to a number of delivery points, *Operations Research Quarterly* **23**, 333-344.

Zbigniew, J. C.and Piotr C. (2002). Parallel Simulated Annealing for the Vehicle Routing Problem With Time Windows. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands–Spain.