DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EXACT AND HEURISTIC ALGORITHMS FOR THE VARIANTS OF THE VEHICLE ROUTING PROBLEM

by

Pınar MIZRAK ÖZFIRAT

November, 2008

İZMİR

EXACT AND HEURISTIC ALGORITHMS FOR THE VARIANTS OF THE VEHICLE ROUTING PROBLEM

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program

by
Pınar MIZRAK ÖZFIRAT

November, 2008
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **“EXACT AND HEURISTIC ALGORITHMS FOR THE VARIANT OF THE VEHICLE ROUTING PROBLEM”** completed by **PINAR MIZRAK ÖZFIRAT** under supervision of **Prof. Dr. HASAN ESKİ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Hasan ESKİ

Supervisor

Prof. Dr. Miraç BAYHAN	Prof. Dr. Tatyana YAKHNO
Thesis Committee Member	Thesis Committee Member
Prof. Dr. İrem ÖZKARAHAN	
Second Supervisor	Examining Committee Member
Examining Committee Member	Examining Committee Member

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

Firstly, I would like to express my deepest gratitude to my advisor, Prof. Dr. İrem Özkarahan for her continous support, guidance and patience throughout my PhD. This dissertation could not have been written without Prof. Dr. İrem Özkarahan who is a dedicated and encouraging advisor. Also, I would like to thank to my second advisor, Prof. Dr. Hasan Eski who helped me throughout the progress of this dissertation.

I would also like to thank to my committee members Prof. Dr. Miraç Bayhan and Prof. Dr. Tatyana Yakhno for their helpful comments and advice. In addition, I would like to say that I am grateful to all my instructors and professors in Middle East Technical University and Dokuz Eylul University for them equipping me with their knowledge and academic skills.

I also want to specially thank to my friends, Rahime Sancar Edis, Özlem Uzun Araz, Emrah Edis and Ceyhun Araz for their friendship, encouragement and support.

Special thanks to my parents, Eser and Yavuz Mızrak, and my brother, Çınar Mızrak, for their love, support and understanding. Finally, I would like to express my special gratitude to my husband, M. Kemal Özfırat, for his love, encouragement, endless patience and support throughout all my PhD years and to my son Ege Özfırat for him not creating me any trouble during the writing process of this dissertation.

Pınar MIZRAK ÖZFIRAT

EXACT AND HEURISTIC ALGORITHMS FOR THE VARIANTS OF THE VEHICLE ROUTING PROBLEM

ABSTRACT

As the world is globalizing, distribution of goods and services becomes an inevitable part of both trade and daily life. Distribution of goods and services from a supply point to various demand points is called logistics. A complete logistics system includes transporting materials from a number of suppliers to the factory plant for manufacturing, transporting the products to warehouses and finally distributing them to the customers. Both the supply and distribution procedures require effective transportation planning. Good transportation planning can save a company a considerable amount of its total distribution costs.

Vehicle Routing Problem (VRP) basically considers transportation planning and has received a lot of attention in operations research literature due to its commercial value. VRP consists of designing m vehicle routes to minimize total cost, each starting and ending at the depot such that each customer is visited exactly once. Since VRP was first introduced in literature, many variations have appeared by including additional assumptions into the problem.

In this dissertation, three of the variants of VRP, which are faced quite often in real life distribution problems, are considered. These are heterogeneous VRP (HVRP), split delivery VRP (SDVRP) and VRP with time windows (VRPTW). A novel Threshold Algorithm is developed for HVRP, SDVRP and small scale VRPTW. For large scale VRPTW, a SetCovering Algorithm is developed.

In order to see the efficiency and performance of these algorithms, they are tested on the literature benchmark problems. The results of the computational experiments indicate that the proposed methodologies are useful tools especially for large scale real life problems where fast decision making is of crucial importance. In addition to performance tests, the proposed methodologies are employed to solve the real life fresh goods distribution problem of a retail chain store. The results achieved are presented to the firm and new distribution strategies are offered.

Keywords : Heterogeneous Vehicle Routing Problem, Split Delivery Vehicle Routing Problem, Vehicle Routing Problem with Time Windows, Interactive Fuzzy Goal Programming, Constraint Programming.

ARAÇ ROTALAMA PROBLEMİ TİPLERİ İÇİN KESİN VE SEZGİSEL ALGORİTMALAR

ÖZ

Dünyanın globalleşmesi ile, ürünlerin ve hizmetlerin dağıtımı hem ticaretin hem de günlük hayatın kaçınılmaz bir parçası haline gelmiştir. Ürünlerin ve hizmetlerin dağıtımına kısaca lojistik denilebilir. Bütün bir lojistik sistemi, malzemeleri tedarikçilerden fabrika binasına üretime ya da işlenmeye götürmeyi, ardından ürünleri depolara taşımayı, ve son olarak da depolardan müşterilere ulaştırmayı kapsamaktadır. Hem tedarik hem de dağıtım işlemleri etkili taşıma planlamasını gerektirir. İyi bir taşıma planı, firmaların toplam dağıtım maliyetlerinin önemli bir kısmını azaltabilir.

Araç Rotalama Problemi (ARP) temel olarak dağıtım planlaması ile ilgilenir ve ticari önemi sayesinde yöneylem araştırması literatüründe çok ilgi toplamıştır. ARP toplam dağıtım maliyetlerini minimize etmek amacıyla, tümü depoda başlayıp depoda biten ve her müşteriye sadece bir defa uğrayan m adet rota tasarlama işleminde kullanılır. ARP literatürde ilk tanımlandığından bu yana, probleme çeşitli varsayımlar eklenerek birçok değişik tipi elde edilmiştir.

Bu tez çalışmasında, gerçek yaşam dağıtım problemlerinde sıkça karşılaşılan ARP'nin üç farklı tipi ele alınmıştır. Bunlar sırasıyla, heterojen filolu ARP (HARP), bölünmüş dağıtımlı ARP (BDARP) ve zaman pencereli ARP'dir (ZPARP). HARP, BDARP ve küçük ölçekli ZPARP için yeni bir Eşik Algoritması geliştirilmiştir. Büyük ölçekli ZPARP için ise yine orjinal olan KümeKaplama Algoritması geliştirilmiştir.

Bu algoritmaların verimliliğini ve performansını ölçmek için, literatürde bulunan test problemleri üzerinde deneyler yapılmıştır. Elde edilen sonuçlar önerilen algoritmaların özellikle hızlı karar vermenin çok önemli olduğu problemlerde faydalı olabileceğini göstermiştir. Literatür deneylerine ek olarak, geliştirilen algoritmalar bir market zincirinin taze gıda dağıtımı problemine uygulanmıştır. Elde edilen sonuçlar firmaya sunulmuş ve yeni dağıtım stratejileri önerilmiştir.

Anahtar Kelimeler: Heterojen Filolu Araç Rotalama Problemi, Bölünmüş Dağıtımlı Araç Rotalama Problemi, Zaman Pencereli Araç Rotalama Problemi, Interaktif Bulanık Hedef Programlama, Kısıt Programlama.

CONTENTS

CHAPTER ONE
INTRODUCTION

Logistics may be defined as the distribution of goods and services from a supply point or supply points to various demand points. A complete logistics system includes transporting materials from a number of suppliers to the factory plant for manufacturing or processing, transporting the products to warehouses or depots and finally distributing them to the customers. Both the supply and distribution procedures require effective transportation planning. Good transportation planning can save a company a considerable amount of its total distribution costs.

One of the major items in the total distribution costs is the traveling and fixed costs of distribution vehicles. Transportation management, and more specifically vehicle routing, has a considerable economical impact on all logistic systems. Effective vehicle routing can save a considerable amount of distribution costs for the company. Optimization of routes for vehicles given various constraints is the origin of vehicle routing problem (VRP). In a practical aspect, this problem contributes directly to a real opportunity to reduce costs in the area of logistics.

The standard version of the VRP which is called the capacitated VRP (CVRP) is easy to state and very difficult to solve. The problem is to generate a sequence of deliveries for each vehicle in a homogeneous fleet based at a single depot so that all customers are serviced and the total distance traveled by the fleet is minimized. Each vehicle has a fixed capacity and must leave from and return to the depot. Each customer has a known demand and is serviced by exactly one visit of a single vehicle. The standard vehicle routing problem was introduced in the operations research and management science literature about 50 years ago. Since then, the vehicle routing problem has attracted an enormous amount of research attention.

In the following section of this chapter, the background of VRP and the motivation behind this dissertation is given. The research objectives and the original contributions are listed in Section 1.2. Finally, in Section 1.3, the organization of this dissertation is outlined.

1.1 Background and Motivation

In 1959, a paper by Dantzig and Ramser appeared in the Journal of Management Science concerning the routing of a fleet of gasoline delivery trucks between a bulk terminal and a number of service stations supplied by the terminal. The distance between any two locations is given and a demand for a certain product is specified for the service stations. The problem is to assign service stations to trucks such that all station demands are satisfied and total mileage covered by the fleet is minimized. The authors imposed the additional conditions that each service station is visited by exactly one truck and that the total demand of the stations supplied by a certain truck does not exceed the capacity of the truck. The problem formulated in the paper by Dantzig and Ramser was given the name "truck dispatching problem". Later on, this problem is referred as the vehicle routing problem and this name has caught on in the literature.

Today, in the globalizing world of technology, transportation has become an inevitable part of both daily life and business life. Since in every type of transportation problem, time and budget constraints appear, effective scheduling of vehicles can only be made by scientific methods. Therefore assignment of vehicles to routes has become a major interest of many researchers due to its relevance in practice.

Designing the optimal set of routes for a fleet of vehicles in order to serve a given set of customers is simply called VRP. It is one of the hardest combinatorial optimization problems. There are many variations of VRP both in real life applications and literature studies.

CVRP which is the classical version of VRP is the one which has found interest most widely in literature. In CVRP, the vehicle fleet is homogeneous. That is, all vehicles are identical in capacity and cost. However, in real life problems, this is not the case most of the time. There may exist different types of vehicles with different capacities in the vehicle fleet. Also these vehicles may have different fixed and variable traveling costs. When this is the case, the problem is formulated as a heterogeneous VRP (HVRP). HVRP is a variation of CVRP in which there is a heterogeneous fleet of vehicles for distribution. Although, heterogeneous vehicle fleet assumption is more realistic, HVRP has not received much attention in the literature. This may be due to the fact that it is more challenging to solve.

One common assumption in CVRP and HVRP is that a customer demand must be satisfied by one visit of a vehicle. In other words, splitting the delivery of a customer among vehicles is not allowed. However, split delivery distribution strategy may decrease distribution costs since it is a relaxation of non-split delivery strategy. Therefore, the effect of incorporating split deliveries into VRP should be studied profoundly. When split deliveries are allowed in VRP, the problem is called split delivery VRP (SDVRP).

In addition to HVRP and SDVRP, another important variation of VRP is VRP with time windows (VRPTW). VRPTW is a well-known and complex combinatorial optimization problem where nodes to be visited require specified time intervals for the visit. VRPTW has started to gain attention in literature recently. This is because as competition increases in business markets, customer requests become more important. In order to gain a competitive edge, companies should deliver the products within requested time intervals of customers. Many of the earlier exact and heuristic methods developed for CVRP has also been applied to VRPTW. However, heuristic methods do not perform well due to their limitations in the search space. Exact methods on the other hand are able to solve small size problems but inefficient for large scale problems.

The main motivations behind this research are the discussions given above. In the light of these discussions, finding efficient and effective solutions for HVRP, SDVRP and VRPTW constitute the main objective of this dissertation.

1.2 Research Objectives and Original Contributions

In this dissertation, a novel Threshold Algorithm is developed which is a cluster first route second type of algorithm (Laporte and Semet, 2002). Both the clustering phase and the routing phase of the algorithm employs advanced tools and has special design characteristics for each of the variations of VRP considered; HVRP, SDVRP and VRPTW. The objectives of the research and the novelties proposed to achieve these objectives are given in the following.

- The main challenge in all types of VRPs lies in the subtour elimination constraints. A subtour is the tour of a vehicle without visiting the depot node which is undesirable (Eg. The route of a vehicle Node 1-Node 2-Node 1 is a subtour). If subtours are not considered, VRP can be solved through mathematical modeling to optimum. However, insertion of subtour constraints makes the problem np-hard.

  In this research, a novel subtour elimination algorithm (SEA) is proposed. SEA is an iterative algorithm which is integrated into a mathematical model. It completely removes subtour constraints from the mathematical model at the very first iteration. Then, it adds only the anticipated subtour constraints at each iteration. By this way, the model can be solved to optimum.

- In HVRP, there is one more point that makes the problem even more challenging. That is as the vehicles with different capacities and fixed costs are included in the problem, a tradeoff between total fixed costs and total traveling costs arises.

As the capacity of a vehicle increases, its fixed cost also increases in HVRP. That is, vehicles in larger capacity are more expensive than smaller ones. Therefore, if larger capacity vehicles are selected, fixed costs increase. In this case, number of tours visiting the depot is smaller and hence traveling costs decrease. On the other hand, if smaller capacity vehicles are selected, fixed costs decrease but traveling costs increase. Therefore the conflict between these two cost terms should be worked out.

When there exist multiple objectives in a problem and the degree of fulfillment of these objectives is vague, fuzzy mathematical programming may be a useful tool to handle the problem (Zimmerman, 1978). Therefore in this dissertation, an interactive fuzzy goal programming (IFGP) approach is designed to deal with the heterogeneous vehicle fleet. To the best of our knowledge, this is the first study that incorporates fuzzy goal programming into HVRP.

- SDVRP is a relaxation of CVRP since it relaxes the assumption that “each node should be visited by only one vehicle”. However, when this constraint is released, the search space becomes larger. Though it becomes harder to find the optimum solution or an efficient solution.

  In order to deal with this challenge, the SEA developed for HVRP is modified according to split delivery distribution strategy. By this way, the size of the problem is decreased and though it becomes easier to find solutions.

- Due to the special characteristics of VRPTWs, these problems cannot be split into clusters. Therefore, it is necessary to handle VRPTW under two categories. VRPTW, which is already in clusters (small scale), and large scale VRPTW.

  The Threshold Algorithm employed for the other variations of VRP can only be used for clustered VRPTW. For large scale VRPTW, an original SetCovering

Algorithm is developed in this dissertation, which is able to find optimum solutions.

In summary, within this dissertation four original methodologies are developed in order to deal with the four challenges defined above. The outline of the research can be seen in Figure 1.1.

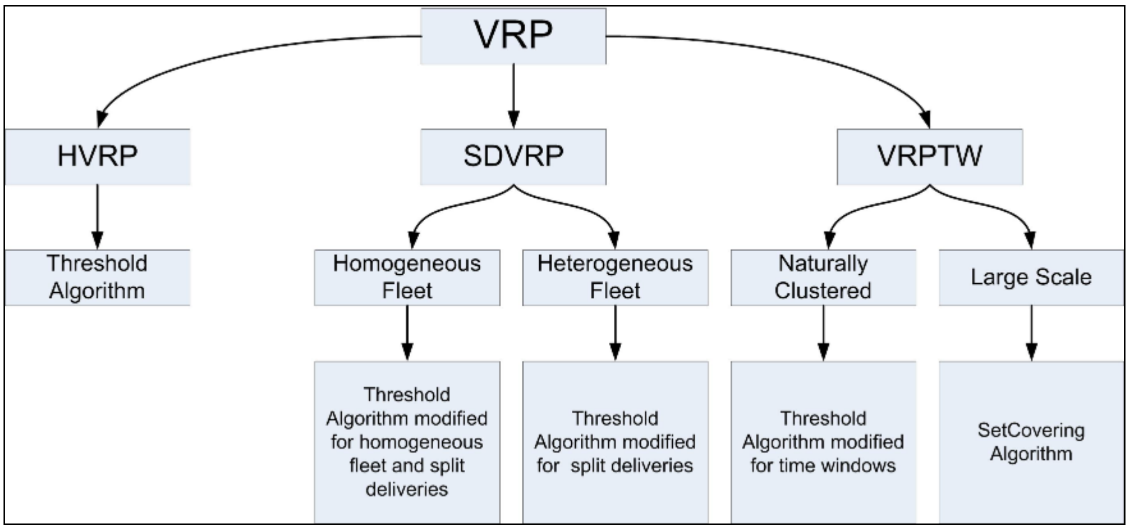

VRP
HVRP
SDVRP
VRPTW
Threshold Algorithm
Homogeneous Fleet
Heterogeneous Fleet
Naturally Clustered
Large Scale
Threshold Algorithm modified for homogeneous fleet and split deliveries
Threshold Algorithm modified for split deliveries
Threshold Algorithm modified for time windows
SetCovering Algorithm

Figure 1.1 Study frame of the dissertation

1.3 Organization of the Thesis

The following chapters of the dissertation are organized as follows.

In Chapter 2, firstly, VRP and its variations are defined. In addition, an overview of solution approaches employed for CVRP is provided. Then a detailed literature review is provided on HVRP, SDVRP and VRPTW, which are the main concerns of this dissertation.

In Chapter 3, the tools employed in the algorithms developed in this dissertation are reviewed comprehensively. Firstly, basics of constraint programming methodology and application are explained. Then, a brief overview on fuzzy sets and fuzzy numbers is given. Finally, IFGP approach, which is one of the fuzzy mathematical modeling techniques, is described in this chapter.

Chapter 4 presents the novel Threshold Algorithm which is designed for HVRP in this dissertation. Also, performance tests on the benchmark instances from the literature are given. In addition, the algorithm is applied to the real life fresh goods distribution problem of a retail chain store. The case, application and the solutions are also provided in this chapter.

In Chapter 5, SDVRP is handled under two states. First one assumes a homogeneous vehicle fleet. The second one assumes a heterogeneous fleet. In other words, the second case combines HVRP with split deliveries. The Threshold Algorithm is modified according to both cases and the performance tests are carried out. Lastly, split delivery strategy is employed for the real life distribution problem handled in Chapter 4.

Chapter 6 is devoted to VRPTW. VRPTW is again considered under two cases, which are clustered problems and large scale problems respectively. For clustered problems, the Threshold Algorithm is modified according to time window assumption. For large scale problems, a novel SetCovering Algorithm is developed. Similar to other problems, tests on VRPTW are given in this chapter.

Finally, Chapter 7 includes the concluding remarks of this dissertation and identifies future directions of research.

CHAPTER TWO
LITERATURE REVIEW ON THE VEHICLE ROUTING PROBLEM

The main focus of this dissertation is the distribution of goods between depots and final users. These problems are generally known as the Vehicle routing problems (VRP) in literature. VRP has received a lot of attention in the Operations Research (OR) literature for its commercial value. Basically, it consists of designing a set of vehicle routes, each performed by a single vehicle starting and ending at its own depot such that all requirements of customers and all the operational constraints are satisfied, and the total transportation cost is minimized. The two main elements of this problem are the customers to be visited and the vehicles to perform the visits. These elements have some common characteristics in all VRPs.

Common characteristics of customers are given in the following:

- Each customer is located at a vertex on the road graph.
- Each customer requires an amount of goods (demand), which must be delivered or collected.
- Some customers may require the delivery to be at a certain period of the day.
- There exists a service time required to load or unload the goods at the customer location.

Distribution of goods is performed by a fleet of vehicles whose composition and size can be fixed or variable. Common characteristics of vehicles are given in the following:

- Each vehicle has an originating depot and there is a possibility that its route may end in another depot than the originating one.
- Each vehicle has a capacity restriction which expresses the maximum amount of goods it can load.
- There exists a cost associated with utilization of the vehicle.

Evaluation of the global costs of the routes needs knowledge of the travel cost or travel time between each pair of customers and between the depots and the customers. Each customer and depot is located at a vertex on the road graph. For each pair of vertices ***i*** and ***j***, and arc (***i***, ***j***) is defined whose distance d_{ij} is the distance of the shortest path from node ***i*** to node ***j*** and the corresponding cost is c_{ij}. Also, for each arc (***i***, ***j***), a travel time t_{ij} is defined which is the traveling time of the shortest path from node ***i*** to node ***j***. Several objectives can be considered for the VRPs. These objectives are usually contrasting with each other. Some of them can be listed as:

- minimization of the total transportation cost which depends on the total distance traveled and/or fixed costs of vehicles in use,
- minimization of the number of vehicles required to serve all customers,
- balancing the routes for travel time and vehicle load,
- minimization of the penalties associated with partial service of the customers,
- or any weighted combination of these objectives.

VRP was first introduced by Dantzig and Ramser in 1959 as the truck dispatching problem. The authors proposed the first mathematical programming formulation and algorithmic approach for the solution of the problem. A few years later, Clark and Wright developed a greedy heuristic which improved Dantzig and Ramser's approach (Toth, Vigo, 2002). Since then, many researchers have dedicated their researches to develop efficient algorithms for dealing with VRP and its extensions. Many models, exact and heuristic approaches have been proposed to find optimal and/or approximate solutions to VRPs.

The very basic version of VRP is called the Capacitated VRP (CVRP) in which demands of customers are deterministic and may not be split; all vehicles are identical in capacity and cost. The objective is to find a number of vehicle routes to minimize cost such that

(i) each route visits the depot node,

(ii) each customer node is visited exactly by one route,

(iii) the sum of the demands on a route does not exceed the capacity of the vehicle.

By incorporating different assumptions to CVRP, variations of VRP are created. Some of the variants of VRP, which have been studied in literature, are listed below.

- Distance Constrained VRP (DCVRP): In DCVRP, in addition to CVRP assumptions, a maximum route length or maximum route time is incorporated. That is the vehicles should complete their routes within a specified time or the routes should not exceed a specified length.
- VRP with Time Windows (VRPTW): VRPTW is an extension of CVRP where a time interval $[\boldsymbol{e_i},\boldsymbol{l_i}]$ and a service time $\boldsymbol{s_i}$ is associated with each customer. In addition to CVRP constraints, each customer should be visited within its specified time window ($[\boldsymbol{e_i},\boldsymbol{l_i}]$) and the vehicle stops at node $\boldsymbol{i}$ for $\boldsymbol{s_i}$ time units to complete service.
- VRP with Backhauls (VRPB): In VRPB, the customer set is divided into two groups. The first subset, $\boldsymbol{L}$, contains $\boldsymbol{n}$ linehaul customers, which require a given quantiy of product to be delivered. The second subset, $\boldsymbol{B}$, contains $\boldsymbol{m}$ backhaul customers, which require a given quantity of products to be picked up. Customers are numbered so that $\boldsymbol{L=\{1, 2,\dots, n\}}$ and $\boldsymbol{B=\{n+1, n+2,\dots, n+m\}}$. The assumption, which regulates the service to these customers, state that if a route contains linehaul and backhaul customers, all linehaul customers must be served before backhaul customers. So, in addition to CVRP constraints,

  (i) in each route, linehaul customers precede backhaul customers,

  (ii) the total demands of linehaul and backhaul customers on a route should not exceed, seperately, the vehicle capacity.
- VRP with Pickup and Delivery (VRPPD): In VRPPD, two quantities, $\boldsymbol{d_i}$ and $\boldsymbol{p_i}$, are associated with each customer $\boldsymbol{i}$ stating the quantity of products to be delivered to customer $\boldsymbol{i}$ and picked up from customer $\boldsymbol{i}$ respectively. That

means there is a net demand of d_i-p_i at customer i. Though it may be positive or negative. Also, for each customer i, O_i denotes the node which is the origin of the delivery quantity and D_i denotes the node which is destination of the pick up quantity. That is the delivery quantity to node i is picked up from O_i, and the quantity picked up from node i is delivered to D_i. Therefore in addition to CVRP constraints,

(i) the load of the vehicle at any time must be nonnegative and must be not exceed the vehicle capacity,

(ii) for each customer i, customer O_i, if different from the depot node, must be served on the same route before customer i,

(iii) for each customer i, customer D_i, if different from the depot node, must be served on the same route after customer i,

- Heterogeneous VRP (HVRP): In HVRP, there exists a heterogeneous fleet of vehicles. That is the capacities and the costs of vehicles may be different for each vehicle. Therefore, the total demand of a route should not exceed the capacity of the vehicle scheduled to that route.
- Stochastic VRP (SVRP): SVRP is the extension of CVRP where some components of the problem are stochastic. For example, travel times or demand quantities or customer nodes may be stochastic.
- Split Delivery VRP (SDVRP): SDVRP can be considered as a relaxation of CVRP. In CVRP a customer can only be visited by exactly one vehicle (as long as the demand does not exceed the capacity of the vehicle). On the other hand, in SDVRP the deliveries of a customer can be split between two or more vehicles. In other words a customer node can be visited by more than one vehicle.

All of the extensions of VRP listed above are directly derived from CVRP. In addition to these variants, there exist other studies in the literature which incorporate more than one assumption at the same time. In other words, intersections of these extensions are also studied. For example by assuming backhauling and time windows together, VRP with backhauls and time windows (VRPBTW) is derived. In Figure 2.1, the direct extensions of CVRP and their interconnections can be seen. In the figure, the ones which are well known in the literature are listed. However, the trend in VRP is to incorporate more assumptions on the same problem. That is, there will be more interconnections which will be handled in the literature in the future. In Figure 2.1, there is one more thing to be pointed. For the extensions given in region I, popular and widely used test instances exist in the literature. The studies in this region mostly concern these test instances. However, in region II, real life applications take place since there are not well known test instances for these type of problems.

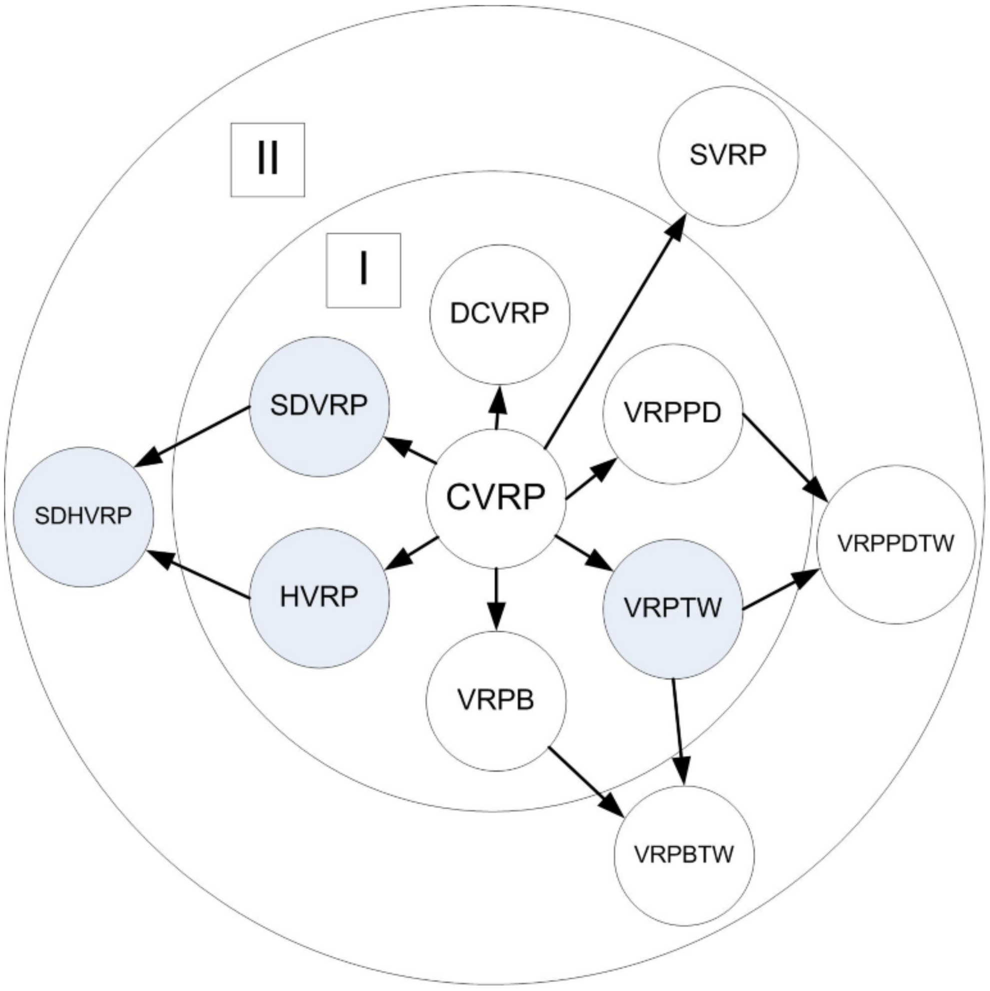


Figure 2.1 Extensions of CVRP and their interconnections

In this dissertation, the VRPs which are shaded in Figure 2.1 are handled. These are HVRP, SDVRP and VRPTW from region I. Also, VRP which allows split deliveries and heterogeneous fleet together is considered (from region II).

Firstly, HVRP is taken into consideration. In real life distribution problems, most of the companies own a heterogeneous fleet of vehicles. Therefore, heterogeneous vehicle fleet assumption is more realistic for most cases. However, HVRP is not studied much in the literature. The reason to this situation may be because it incorporates more restrictions into VRP. The main conflict in HVRP arises from the tradeoff between vehicle fixed costs and total traveling costs. As the capacity of vehicles decrease, fixed costs also decrease. But in this case, number of required vehicles increases. Hence, total traveling cost increases. On the other hand, as larger capacity vehicles are selected, number of required vehicles and hence total traveling cost decreases. But this time, fixed costs increase. The tradeoff between these two terms should be worked out to find an efficient solution to HVRP. Therefore, it cannot be handled with the approaches developed for CVRP. HVRP requires specially designed algorithms. In short, it can be said that due to

- its realistic assumptions
- limited interest in literature
- the challenge between vehicle fixed costs and traveling costs

HVRP is firstly considered in this dissertation.

SDVRP is the second extension of VRP which is considered in this research. When the demand of a customer exceeds the capacity of vehicles, allowing split delivery is a must. Other than this case, allowing split deliveries is determination of a distribution strategy. Therefore, in a distribution problem, both distribution strategies (allowing split deliveries or allowing non-split deliveries) should be compared in terms of distribution objectives. One of them should be selected according to the results. Allowing split delivery strategy can be considered as a relaxation of non-split delivery strategy. Therefore, it is likely that distribution costs would decrease under split delivery assumption. But still, the research in this area is very limited in the literature. There are very few studies considering SDVRP. So, due to the

- need to compare the two distribution strategies
- limited reaserch in this area

SDVRP is handled in this dissertation.

Another variation of VRP considered is VRPTW. VRPTW started to gain attention since the beginning of 90s. This is due to the increase in competition in service industry. As competition increases, distributors started taking into care not only the demand quantities of customers but also the delivery requirements. Therefore, time windows of deliveries started to be more important. Hence, VRPTW became the interest of researchers. Although there are quite many studies in this area, to the best of our knowledge, there is no exact algorithm for VRPTW. Still, many researchers are studying VRPTW to develop effective and efficient algorithms. Therefore, due to the

- increase in VRPTW applications in real life
- need for effective and efficient algorithms in this area

VRPTW is handled in this dissertation.

In addition to these three main extensions of VRP, distribution which handles both heterogeneous fleet of vehicles and split deliveries is considered. In other words, split delivery heterogeneous VRP (SDHVRP) is studied. To the best of our knowledge, there are no studies in the literature which merges these two assumptions together. So by SDHVRP, a distribution strategy (allowing split deliveries or not) can be determined for HVRP.

In order to be able to design effective algorithms for these extensions of VRP, firstly the very basic version, which is CVRP, should be understood and the research carried out in this area should be studied. Then the literature on HVRP, SDVRP and VRPTW should be examined. Sections 2.1, 2.2, 2.3 and 2.4 are devoted to these subjects respectively.

2.1 Capacitated Vehicle Routing Problem

CVRP can be simply stated as the problem of determining optimal routes through a set of locations and defined on a directed graph $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{A})$ where $\boldsymbol{N} = (\boldsymbol{n_0}, \boldsymbol{n_1}, \ldots, \boldsymbol{n_n})$ is a vertex set and $\boldsymbol{A} = ((\boldsymbol{n_i}, \boldsymbol{n_j}) : \boldsymbol{n_i}, \boldsymbol{n_j} \in \boldsymbol{N}, i \neq j)$ is an arc set. Vertex $\boldsymbol{n_0}$ represents a depot node where a fleet $\boldsymbol{V} = (\boldsymbol{v_1}, \ldots, \boldsymbol{v_n})$ of vehicles exist with an identical and uniform capacity $\boldsymbol{Q}$. All remaining vertices represent customers. A non-negative (distance/cost) matrix $\boldsymbol{C} = (\boldsymbol{c_{ij}})$ is defined on $\boldsymbol{A}$. A non-negative weight $\boldsymbol{d_i}$ is associated with each vertex to represent the customer demand at $\boldsymbol{n_i}$, and the total demand assigned to any route may not exceed the vehicle capacity $\boldsymbol{Q}$. Thus, CVRP aims at determining vehicle routes of minimal total cost, each starting and ending at the depot, so that every customer is visited exactly once. A typical mathematical formulation for the single depot CVRP is given in the following where $\boldsymbol{X_{ijv}}$ is a binary decision variable indicating whether vehicle $\boldsymbol{v}$ goes from $\boldsymbol{n_i}$ to $\boldsymbol{n_j}$.

$$\left.\begin{array}{lll}
Minimize \sum_i \sum_j \sum_v X_{ijv} * c_{ij} & & (2.1a) \\
Subject\ to & & \\
\sum_j \sum_v X_{ijv} = 1 & \forall i \in N & (2.1b) \\
\sum_i \sum_v X_{ijv} = 1 & \forall j \in N & (2.1c) \\
\sum_i X_{i0v} \leq 1 & \forall v \in V & (2.1d) \\
\sum_j X_{0jv} \leq 1 & \forall v \in V & (2.1e) \\
\sum_i X_{ikv} = \sum_j X_{kjv} & \forall k \in N, \forall v \in V & (2.1f) \\
\sum_i \left(\sum_j X_{ijv} \right) * d_i \leq Q & \forall v \in V & (2.1g) \\
X_{ijv} \in Z & \forall i, j \in N, \forall v \in V & (2.1h) \\
X_{ijv} \in \{0,1\} & \forall i, j \in N, \forall v \in V & (2.1i)
\end{array}\right\} \quad \textbf{(2.1)}$$

In the objective function of Formulation 2.1 (Equation 2.1a), the total distance traveled is minimized. By constraints ***(2.1b)*** and ***(2.1c)***, each node is visited exactly once. Constraint ***(2.1d)*** and ***(2.1e)*** state that every vehicle must go out of and into the depot node. Constraints ***(2.1f)*** assure that a vehicle ingoing to a node must leave that node. ***(2.1g)*** states that the capacities of vehicles should not be exceeded. Constraint set ***(2.1h)*** eliminates subtours where,

$$Z = \left\{ (X_{ijv}) : \sum_{i \in B} \sum_{j \in B} X_{ijv} \le |B| - 1, \forall B \subseteq V / \{0\}; |B| \ge 2 \right\} \qquad \textbf{(2.2)}$$

Finally, constraints ***(2.1i)*** are cardinality constraints.

2.1.1 Optimization Approaches Applied to CVRP

CVRP is one of the most studied versions of VRP in the literature. Both exact and heuristic approaches exist for CVRP. The most popular approach, which solves CVRP to optimality, is branch and bound. However, the size of the problems, which can be solved optimally, is restricted with a couple of tens of vertices (at most 50 vertices). Some of the older relaxations are based on assignment problem (Laporte et.al., 1986) and minimum spanning tree (Christofides et. al., 1981, Fisher, 1994). In addition to these there are some more recently used bounding approaches. One of these is the additive approach (Fischetti and Toth, 1989), which allows different bounding procedures. Fischetti et. al. (1994) describes two relaxations using additive approach. The first one is based on disjunction of infeasible arc sets and the second one is based on minimum cost flow respectively (Toth and Vigo, 2002).

Langrangian lower bounds are also recently proposed, sophisticated bounds which allow us to solve larger problems in size. For example, Fisher was able to solve the symmetric CVRP problem with 100 or less customers with 99% close to optimality using a Langrangian lower bound (1994) (Toth and Vigo, 2002).

Although branch and bound is an often used method, it has some drawbacks. If the problem consists of a large number of linear constraints, branch and bound cannot be employed. This is because such a large constraint system cannot be fed into an LP solver. In this case, branch and cut approach is used.

The research on application of branch and cut on VRP is quite limited. Some of the studies in this area belong to Achuthan et. al. (1997), Augerat et al.(1998), Blasum and Hochstattler (2000), Ralphs (2003), Longo et. al. (2006).

Among these studies, Augerat et al. (1998) handled VRP's with upto 135 customers. The authors first separated capacity constraints using tabu search. Then they applied branch and cut procedure. Very few of the instances could be solved optimally. Others obtained very close solutions to optimal.

Blasum and Hochstattler (2000) modified the branching tree and separation procedure of Augerat et al. (1998). They were able to decrease computation time in a recognizable way (Naddef and Rinaldi, 2002).

Ralphs (2003) described a parallel procedure for branch and cut algorithms. The author tested the procedure on instances with 100 customers and has shown that optimal solution could be achieved for some of the problems. It is seen that branch and cut algorithms provide quite good results which is encouraging for future research. However, it is still very limited for large scale problems.

Another attempt to solve CVRP optimally is done by Ghiani and Improta (2000). They transformed the problem into a capacitated arc routing problem for which an exact algorithm and several approximate algorithms exist in the literature. When the exact algorithms are incapable of handling VRP, such as in very large scale problem sizes, heuristic approaches are employed. Heuristic approaches are practical, fast and provide not optimal but good solutions. Heuristics can be split into two categories as classical, developed mostly between 1960 and 1990, and metaheuristics, developing and growing since early 90's.

2.1.2 Classical Heuristic Approaches Applied to CVRP

There are quite a number of studies handling VRP by heuristic approaches in the literature. Some of them are listed here.

Classical heuristic approaches are divided into three: Constructive heuristics, two-phase heuristics, and improvement methods. Constructive heuristics build a feasible solution step by step while tracking the objective value. In two phase heuristics, clustering and routing phases appear either in a "cluster first - route second" or "route first - cluster second" way. Finally improvement heuristics try to upgrade the quality of a feasible solution by exchanging edges or vertices.

Clark and Wright (1964) Savings Algorithm is the oldest and most known heuristic for CVRP. It is a constructive heuristic based on computation of savings of each possible edge and inserting the one with the best saving into the route. This algorithm has become the basis of many studies as Gaskell (1967), Yellow (1970), Golden et. al. (1977), Paessens (1988), Nelson et al.(1985) (Laporte and Semet, 2002). Also different modifications of Clark and Wright algorithm are developed by Desrochers and Verhoog (1991) and Wark and Holt (1994) (Laporte and Semet, 2002).

Among the constructive heuristics, there are sequential insertion heuristics in the literature. Two of these are known as Mole and Jameson (1976) algorithm and Christofides algorithm (Christofides et. al., 1979) (Laporte and Semet, 2002).

When two phase algorithms are considered, "cluster first - route second" version took more attention then "route first - cluster second" in the literature. Sweep algorithm developed by Wren in 1971 and Fisher and Jaikumar algorithm (1981) are well known "cluster first - route second" algorithms. An extension of the sweep algorithm is the petal algorithm applied by Gillett and Miller (1974), Foster and Ryan (1976), Agarwal et. al. (1989), Ryan et. al. (1993), Renaud et. al. (1996) (Laporte and Semet, 2002).

"Route first cluster second" algorithm was first put forward by Beasley (1983). The first phase constructs a giant tour and the second phase becomes a shortest path problem. Haimovich and Kan (1985) showed that this algorithm is optimal when all customers have unit demand but not for general demands (Laporte and Semet, 2002).

All algorithms in "cluster first - route second" set have computational comparisons in the literature. However, to the best of our knowledge, there exists no comparative study for "route first - cluster second" algorithm in the literature.

The improvement heuristics can be handled in two ways: Either considering each vehicle route separately or handling multiple routes at a time. When single route is considered, the problem turns into a traveling salesman problem (TSP). Lin has proposed λ-opt algorithm (1965) for TSP which can also be applied to CVRP. In this algorithm, λ edges are removed from the feasible solution and any other profitable connection is inserted into the route. Most of the improvement heuristics developed base on this mechanism such as Lin and Kerninghan (1973), Or (1976), Renaud et. al. (1996) (Laporte and Semet, 2002).

When multi routes are considered in parallel, edge exchanges between routes are performed. Some of the heuristics base on this idea belong to Thompson and Psanottis (1993), Van Breedam (1994) (Laporte and Semet, 2002), and Kindervater and Savelsbergh (1997).

Classical heuristics provide quick solutions. However, the quality of the solutions are surpassed by those of the metaheuristics. Until today, simulated annealing (SA), tabu search (TS), genetic algorithms (GA), ant systems (AS), and neural networks (NN), have been applied to CVRP.

2.1.3 Metaheuristic Approaches Applied to CVRP

Among the metaheuristic approaches applied to CVRP, TS is the most widely used heuristic and provides the most promising results. Some of the algorithms are developed by Osman (1993), Gendreau et. al. (1994), Taillard (1993), Xu and Kelly (1996) (Gendreau et. al., 2002), Barbarosoglu and Ozgur (1999).

In Osman's algorithms (1993), the neighborhoods are defined based on 2-interchange generation mechanism. Tabu Route algorithm of Gendreau et. al. (1994) allows infeasible solutions in iterations. By this way, the search can be diversified into a broader area. Taillard's algorithm (1993) is based on a distinctive idea. The problem is decomposed into subproblems. Each subproblem is solved independently. After a constant number of iterations, some of the vertices should be moved to adjacent subproblems. Xu and Kelly (1996) consider swaps of vertices between two routes in neighborhood generation. In addition, they try to position some of the vertices optimally by solving a network flow model (Gendreau et. al., 2002). TS algorithm of Barbarosoglu and Ozgur (1999) use λ-interchange in neighborhood generation and give priority to vertices which are far from the center of their current route but close to the center of the new route.

In addition to these studies, Rochat and Taillard (1995) introduced the adaptive memory concept into TS (Adaptive memory property can be incorporated into other metaheuristics too). Adaptive memory is the process of extracting best routes from existing solutions. However, one should be careful not to include same vertices in the new solution (Gendreau et. al., 2002).

Another concept in this area is granular TS (GTS) developed by Toth and Vigo (1998). It is based on the idea that longer edges on a graph are less likely to belong to the optimal solution. Therefore, in GTS, a granularity threshold is set and the edges longer than this threshold are not considered at all. By this way, computation times are decreased considerably (Gendreau et. al., 2002). When all these TS algorithms are compared on the same benchmark instances, it is found that Taillard's algorithm (1993) achieved the highest number of best solutions, and GTS of Toth and Vigo (1998) provided the best computation times (Gendreau et. al., 2002).

Adaptive memory property is also employed by other researchers. One of these studies is done by Tarantilis (2005). The author develops an adaptive memory programming method called "Solutions Elite Parts Search". The method first generates initial solutions and stores these in an adaptive memory. A constructive heuristic merges route components kept in the adaptive memory. Finally TS is used to improve the solution.

Application of other metaheuristic approaches on VRP is rather limited. Osman applied SA to the symmetric version of CVRP (1993). He tested the approach on some benchmark problems. Very few of the solutions were able to provide the value of the best known solution in the literature. Also, it is seen that computation times were quite long.

GA's are thought to be ineffective for VRP since the beginning of 2000's. But two of the recent studies provided competitive results compared to TS for some benchmark studies. One of these studies belong to Baker and Ayechew (2003). A hybrid GA method with neighborhood search method is applied to CVRP in this paper. The other study belongs to Prins (2004). The author proposed a GA without trip deliminiters and which uses a local search procedure.

Ant algorithms have also found application in CVRP by Bullnheimer et. al. (1999) and Mazzeo and Loiseau (2004). The result of these studies were quite close to the best known solutions. The last of the metaheuristics employed to solve CVRP is NN. One of the studies belong to Ghaziri (1996). The algorithm proposed is tested on benchmark problems and the computational results have shown that it produces relatively good solutions but far away from best TS applications (Gendreau et. al., 2002).

Other than these techniques, constraint programming approach is used by Shaw (1998) for CVRP. He proposed a local search method called large neighborhood search. This method explores a large neighborhood by removing some of the edges from the graph and reinserting these using a constraint based tree search. The results are encouraging to apply constraint programming technology to VRP.

An extension of large neighborhood search is presented by Pisinger and Ropke (2007) by adding an adaptive layer to the method. The new algorithm called adaptive large neighborhood search is employed to solve several different versions of VRP including CVRP. The method chooses among a number of insertion and removal heuristics adaptively which provides robustness according to the problem type.

More recently, researchers started to look for solutions to real life problems which are very large scale. In such problems, the main point is to find an efficient solution in an effective time. One of the studies carried out belong to Li et. al. (2005). The authors developed instances with up to 1200 customers. They proposed a method called variable length neighbor list, which is able to reduce some of the unproductive computations.

2.2 Heterogeneous Fleet Vehicle Routing Problem

Heterogeneous VRP (HVRP) is a variation of CVRP in which there is a heterogeneous fleet of vehicles for distribution. In CVRP, all vehicles are assumed to be identical with capacity $\boldsymbol{Q}$. However, in real life problems, this is not the case most of the time. There may exist different types of vehicles with different capacities. Also these vehicles may have different fixed and variable traveling costs. Since HVRP impose more restrictions over CVRP, it is also np-hard. Due to the complexity of the problem, there has been no exact algorithm developed to solve it yet.

HVRP is studied in two different versions in the literature. Some of the researchers make an assumption that there is an unlimited number of vehicles of each type. They try to find the optimal set of vehicles to be scheduled in the problem. This is called the fleet size and mix VRP (FSMVRP). On the other hand, some researchers study the case where there is a fixed vehicle fleet. They try to schedule this fleet of vehicles to the customers in an optimal way. This problem is called heterogeneous fixed fleet VRP (HFVRP). Although HFVRP is more realistic than FSMVRP, it has attracted less attention in the literature.

One of the earliest papers studying HVRP belongs to Golden et. al.(1984). The authors employed several simple heuristics, such as Clark and Wright heuristic, improvement heuristics etc., on FSMVRP. They derived 20 literature problems from Christofides and Eilon (1969) and Clark and Wright (1964) (Golden et. al. 1984). They tested the heuristics on these problems. The problem data is given in Table 2.1. However, coordinates of eight of these 20 problems are hardly available in the literature. The other 12 of the 20 problems are the most widely used HVRP test problems in the literature. Since 1984, many researchers have tested the performance of their algorithms on these problems.

Another earlier study in this area belongs to Ulusoy in 1985. In this study, both FSMVRP and HFVRP cases are considered. A four phase solution procedure is developed. In the first phase, Chinese postman problem is solved and the arcs requiring service are found under the objective of minimizing total distance traveled. By this way, a giant tour is obtained. In the second phase, this giant tour is split into vehicle subtours with respect to the capacity constraints of vehicles. The third phase handles each of the vehicle tours and solves a shortest path problem within the existing demand points. Finally, a post processor is applied in the last phase to improve solutions.

HVRP is also studied by Desrochers and Verhoog (1991). They proposed an approach to find the best composition of vehicles to serve the customers efficiently (FSMVRP). They presented a savings heuristic based on route fusion. At each step, the best route is selected by solving a weighted matching problem. The algorithm is tested on a set of benchmark problems.

Table 2.1 Problem data of the 20 test instances published in Golden et. al. (1984)

Problem Instance	Number of Nodes	Vehicle A		Vehicle B		Vehicle C		Vehicle D		Vehicle E		Vehicle F	
		Capacity	Cost	Capacity	Cost	Capacity	Cost	Capacity	Cost	Capacity	Cost	Capacity	Cost
1	12	15	20	35	50	60	100						
2	12	30	60	40	90	110	300						
3	20	20	20	30	35	40	50	70	120	120	225		
4	20	60	1000	80	1500	150	3000						
5	20	20	20	30	35	40	50	70	120	120	225		
6	20	60	1000	80	1500	150	3000						
7	30	40	150	100	500	140	800	200	1200	300	2000		
8	30	10	15	50	50	150	200	400	600				
9	30	40	30	100	100	140	160	200	240	300	400		
10	30	40	30	100	100	140	160	200	240				
11	30	30	60	80	200	200	700	350	1500				
12	30	30	40	50	80	75	150	120	300	180	500	250	800
13	50	20	20	30	35	40	50	70	120	120	225	200	400
14	50	120	100	160	1500	300	3500						
15	50	50	100	100	250	160	450						
16	50	40	100	80	200	140	400						
17	75	50	25	120	80	200	150	350	320				
18	75	20	10	50	35	100	100	150	180	250	400	400	800
19	100	100	500	200	1200	300	2100						
20	100	60	100	140	300	200	500						

In 1999, a tabu search heuristic is presented for FSMVRP by Gendreau et al. At first step of the Tabu Search which is constructing the initial solution, a generalized insertion heuristic (GENIUS developed by Gendreau et. al., 1992 for traveling salesman problem) is applied. After initialization, main search goes on with neighboring and evaluating the neighbors in terms of the objective function. In this paper, the objective function is the weighted sum of total cost and vehicle overcapacity. In other words, the objective is penalized with a percentage of vehicle overcapacity. By this way, infeasible solutions are allowed during main search diversifying the search into new areas. After termination the best solutions achieved goes through post optimization. In this procedure, the solutions are tried to be improved by exchanging two vertices or changing the fleet. The whole procedure is applied to the 12 test problems from Golden et. al.(1984) and it is compared with other approaches offered for FSMVRP.

Other studies which proposed competitive solution approaches for the 12 of the test problems published in Golden et. al.(1984) are Taillard (1999), Wassan and Osman (2002) and Choi and Tcha (2007).

Taillard (1999) proposed a heuristic column generation approach for FSMVRP. The column generation procedure works iteratively generating columns by tabu search at each iteration. Wassan and Osman (2002) developed new variants of tabu search metaheuristic. These variants use a mix of different components of tabu search, including reactive tabu search, variable neighborhoods and special data memory structures. Choi and Tcha (2007) also proposed a column generation technique for FSMVRP. The authors developed a tight integer programming model and solved its linear relaxation by column generation technique. Then with the tight lower bounds achieved, they employed branch and bound procedure to obtain an integer solution. These two studies provided some of the new best known solutions to the 12 test problems. The best known solutions to the 12 test instances published in Golden et. al. (1984) are given Table 2.2. The studies who achieved these best known solutions are referred in the table as: Taillard (1999) (T); Gendreau et. al.(1999) (GLMT); Wassan and Osman (2002) (WO); Choi and Tcha (2007) (CT).

Table 2.2 Best known solutions to the 12 instances published in Golden et. al. (1984)

Problem No	Number of Nodes	Solution Cost	Authors
3	20	961,03	T; GLMT; WO; CT
4	20	6437,33	T; GLMT; WO; CT
5	20	1007,05	GLMT; WO; CT
6	20	6516,47	T; GLMT; WO; CT
13	50	2406,36	CT
14	50	9119,03	T; GLMT; CT
15	50	2586,37	T; GLMT; WO; CT
16	50	2720,43	CT
17	75	1744,83	CT
18	75	2371,49	CT
19	100	8659,74	WO
20	100	4039,49	CT

Salhi and Sari proposed a multi-level composite heuristic for FSMVRP. Their study considered multiple depots (1997). The heuristic simultaneously allocates customers to depots and determines the best fleet composition for the delivery routes. It is tested on benchmark problems where up to 360 customers and five vehicle types exist. FSMVRP is also addressed by Ochi et al. (1998). The researchers used parallel GA together with scatter search to solve the problem.

Liu and Shen (1999) presented a route construction method for FSMVRP based on several different insertion heuristics. The case also handles time window constraints. The methods are tested on 100 customer problems in the literature and compared within each other.

Recently more sophisticated methods are started to be applied. For example Lima et. al. (2004) proposed a GA hybridized with GENIUS (Gendreau et. al., 1992) and λ-interchange mechanism for FSMVRP.

As stated before, the other version of HVRP is where there is a heterogeneous fixed fleet of vehicles. There exist fewer studies in the literature for HFVRP compared to FSMVRP. One of these is Tarantilis and Kironoudis's study on distribution of perishable foods (2001). They proposed a metaheuristic algorithm in order to solve the VRP of a diary in Athens distributing fresh milk to 299 customers daily. The diary had three different types of vehicles. The objective of the study was to determine the set of customers to be served by each vehicle and the corresponding routes to minimize the total distance traveled. The authors proposed an adaptive threshold accepting algorithm. The difference of the presented algorithm to classical threshold accepting algorithms is that the threshold can be both lowered or raised from one iteration to the other. The developed algorithm is proved to be quite efficient in the paper.

Burchett and Campion applied tabu search to HFVRP in grocery supply industry (2002). Their algorithm is a combination of several other algorithms in the literature. Initial solutions are found by Salhi and Rand's saving values heuristic (1987). Neighborhood scheme is based on Wassan and Osman's algorithm (2002). The problem assumes stochastic customer demands. It is solved for different delivery periods and the results are compared.

Moghaddam et. al. (2006) proposed a linear integer-programming model for the HFVRP. They solved the model using SA hybridized with nearest neighborhood heuristic.

2.3 Split Delivery Vehicle Routing Problem

SDVRP is a relaxation of the classical VRP. In CVRP a customer can only be visited by one vehicle (as long as the demand does not exceed the capacity of the vehicle). On the other hand, in SDVRP the deliveries of a customer can be split between two or more vehicles. However, including this assumption does not make the problem easier to be solved. It is still np-hard.

There are very few studies concerning SDVRP in the literature. These studies consider VRP where all vehicles are identical and allow split delivery assumption. This problem has been first introduced by Dror and Trudeau (1989). The authors have showed the savings that can be achieved by allowing split deliveries (Archetti et al., 2006). Dror et al. (1994) have formulated the problem as an integer linear program. Then branch and bound algorithm is applied with the relaxation of constraints. The authors developed seven main test instances. They applied the proposed procedure on these test problems and their derivations. Archetti et. al. (2006) has proposed three alternative tabu search algorithms for SDVRP and also tested them on the Dror and Trudeau problems. The comparison of Archetti (2006) and Dror and Trudeau (1994) results for these seven problems are given in Table 2.3. The solutions in bold are the best known solutions published for these problems.

Table 2.3 Comparison of Archetti (2006) and Dror and Trudeau (1994) results. Solutions in bold are best known solutions in the literature.

		Dror and Trudeau (1994)	**Archetti et. al. (In Press)**		
			SPLITABU	**SPLITABU-DT**	**FAST-SPLITABU**
Problem	*n*	*z*	*z*	*z*	*z*
1	50	5866899	**5300570**	5335535	5335535
2	75	8948212	8516729	**8495410**	**8495410**
3	100	9022361	8461844	**8356191**	8357361
4	150	11307108	**10621988**	10698369	10883100
5	199	13757272	13678177	**13428515**	13463934
6	120	10842373	10847331	**10560148**	**10560148**
7	100	9517122	**8226045**	8253184	8253184

Frizzel and Giffin (1995) developed three heuristics for SDVRP and tested these on some benchmark problems. Sierksma and Tijssen (1998) have constructed a column generation technique for a real life application of SDVRP (Belenguer et al., 2000). Belenguer et al. (2000) developed an efficient lower bound for SDVRP where the quantities delivered to customers are integer numbers.

SDVRP is formulated as a dynamic program (DP) with infinite number of states and solution spaces by Lee et al. (2002). Ho and Haugland (2004) considered SDVRP with time windows. They presented a tabu search algorithm to solve the problem and analyzed the performance of the approach on problems with 100 distribution points. More recently, Archetti et al. (2008) have studied and identified the distribution environments in which allowing split deliveries are more beneficial. Moghaddam et al. (2007) also studied split deliveries and developed a simulated annealing approach.

2.4 VRP With Time Windows

VRPTW has started to gain more attention since the beginning of 90s. This is mainly due to the fast growth in service industry. To gain a competitive in service industry, distributors should make their planning according to customer requirements including their time window demands. This fact has led researchers to find more efficient solutions to VRPTW.

Many of the studies in this area is carried out to develop a new solution procedure first, and then its performance is tested on the benchmark problems in the literature. Benchmark problems which have been most commonly chosen to be evaluated belong to Solomon (1987). Solomon introduced 56 problems which have 100 nodes. Then 25 node and 50 node instances are derived from these 56 problems which makes up totally 168 test instances. These problems are split into three sets. The first one is made up of clustered problems, which is named as C series. The nodes in the second set, which is called R series, are spread on the X-Y coordinates evenly. The third set called RC series is made up of an intersection of the first two (Cordeau et. al., 2002).

In the literature, there exits both exact and heuristic approaches for these problems. Among the exact approaches, Kohl et. al. (1999) solved 70 of the 87 short horizon problems to optimality. Four additional problems were solved by Larsen (1999). Also six more are solved by Cook and Rich (1999) and Kallehauge et. al., (2000) (Cordeau et. al., 2002).

When the 81 problems in the long horizon set are considered, Larsen (1999) was the first to provide exact solutions to problems in this set. He solved 17 of the problems. Cook and Rich (1999) was able to provide solution to 13 other problems and Kallehauge et. al. (2000) was able to solve 16 more of them to optimality (Cordeau et. al., 2002).

The solutions achieved for C series, R series and RC series are given in Tables 2.4, 2.5 and 2.6 respectively. The authors in the tables are denoted as follows: Kohl et. al. (1999) (KDMSS); Larse (1999) (L); Kallehauge et. al. (2000) (KLM); Cook and Rich (1999) (CR). However, all the solutions obtained in this table are found by multiplying the distances by ten and truncating the result. Hence some routes may not satisfy all time window constraints when real distances are used. Therefore these solutions are optimal solutions with approximate distances or they can be called as approximate optimal solutions.

More recently, Kallehauge et. al.(2006) developed another exact approach. In this study, the authors have considered the Langrangian relaxation and handled it with a stabilized cutting plane algorithm. The algorithm is embedded in a branch and bound search and strong valid inequalities are introduced. By this procedure the authors were able to solve two benchmark problems to optimality which are the largest problems to be solved up to date.

Another exact approach in this area is constraint programming. Aminu and Eglese (2006) have considered the Chinese postman problem with time windows. They have transformed the problem into an equivalent VRPTW and developed a constraint programming model. The results indicated that when the time windows are tight, optimal solutions can be achieved. But as time windows grow wider, constraint programming takes quite long time to find the optimal solution. A detailed research about exact algorithms on VRPTW can be found in Kallehauge (2008).

Table 2.4 Optimal solutions on C series problems.

Problem No	**Solution Cost**	**Authors**	**Problem No**	**Solution Cost**	**Authors**
C101-25	191,3	KDMSS	C201-25	214,7	CR; L
C101-50	362,4	KDMSS	C201-50	360,2	CR; L
C101-100	827,3	KDMSS	C201-100	589,1	CR; KLM
C102-25	190,3	KDMSS	C202-25	214,7	CR; L
C102-50	361,4	KDMSS	C202-50	360,2	CR; KLM
C102-100	827,3	KDMSS	C202-100	589,1	CR; KLM
C103-25	190,3	KDMSS	C203-25	214,7	CR; L
C103-50	361,4	KDMSS	C203-50	359,8	CR; KLM
C103-100	826,3	KDMSS	C203-100	588,7	KLM
C104-25	186,9	KDMSS	C204-25	213,1	CR; KLM
C104-50	358,0	KDMSS	C204-50	350,1	KLM
C104-100	822,9	KDMSS	C204-100	-	
C105-25	191,3	KDMSS	C205-25	214,7	CR; L
C105-50	362,4	KDMSS	C205-50	359,8	CR; KLM
C105-100	827,3	KDMSS	C205-100	586,4	CR; KLM
C106-25	191,3	KDMSS	C206-25	214,7	CR; L
C106-50	362,4	KDMSS	C206-50	359,8	CR; KLM
C106-100	827,3	KDMSS	C206-100	586	CR; KLM
C107-25	191,3	KDMSS	C207-25	214,5	CR; L
C107-50	362,4	KDMSS	C207-50	359,6	CR; KLM
C107-100	827,3	KDMSS	C207-100	585,8	CR; KLM
C108-25	191,3	KDMSS	C208-25	214,5	CR; L
C108-50	362,4	KDMSS	C208-50	350,5	CR; KLM
C108-100	827,3	KDMSS	C208-100	585,8	KLM
C109-25	191,3	KDMSS			
C109-50	362,4	KDMSS			
C109-100	827,3	KDMSS			

Table 2.5 Optimal solutions on R series problems.

Problem No	Solution Cost	Authors	Problem No	Solution Cost	Authors
R101-25	617,1	KDMSS	R201-25	463,3	CR; KLM
R101-50	1044,0	KDMSS	R201-50	791,9	CR; KLM
R101-100	1637,7	KDMSS	R201-100	1143,2	KLM
R102-25	547,1	KDMSS	R202-25	410,5	CR; KLM
R102-50	909,0	KDMSS	R202-50	698,5	CR; KLM
R102-100	1466,6	KDMSS	R202-100		
R103-25	454,6	KDMSS	R203-25	391,4	CR; KLM
R103-50	772,9	KDMSS	R203-50		
R103-100	1208,7	CR; L	R203-100		
R104-25	416,9	KDMSS	R204-25		
R104-50	625,4	KDMSS	R204-50		
R104-100			R204-100		
R105-25	530,5	KDMSS	R205-25	393	CR; KLM
R105-50	899,3	KDMSS	R205-50	690,9	KLM
R105-100	1355,3	KDMSS	R205-100		
R106-25	465,4	KDMSS	R206-25	374,4	KLM
R106-50	793,0	KDMSS	R206-50		
R106-100	1234,6	CR; KLM	R206-100		
R107-25	424,3	KDMSS	R207-25	361,6	KLM
R107-50	711,1	KDMSS	R207-50		
R107-100	1064,6	CR; KLM	R207-100		
R108-25	397,3	KDMSS	R208-25	330,9	CR; KLM
R108-50	617,7	CR; KLM	R208-50		
R108-100			R208-100		
R109-25	441,3	KDMSS	R209-25	370,7	KLM
R109-50	786,8	KDMSS	R209-50		
R109-100	1146,9	CR; KLM	R209-100		
R110-25	444,1	KDMSS	R210-25	404,6	
R110-50	697,0	KDMSS	R210-50		
R110-100	1068,0	CR; KLM	R210-100		
R111-25	428,8	KDMSS	R211-25	350,9	
R111-50	707,2	CR; KLM	R211-50		
R111-100	1048,7	CR; KLM	R211-100		
R112-25	393,0	KDMSS			
R112-50	630,2	CR; KLM			
R112-100					

Table 2.6 Optimal solutions on RC series problems

Problem No	Solution Cost	Authors	Problem No	Solution Cost	Authors
RC101-25	461,1	KDMSS	RC201-25	360,2	CR; L
RC101-50	944,0	KDMSS	RC201-50	684,8	L; KLM
RC101-100	1619,8	KDMSS	RC201-100	1261,8	KLM
RC102-25	351,8	KDMSS	RC202-25	338,0	CR; KLM
RC102-50	822,5	KDMSS	RC202-50	-	
RC102-100	1457,4	CR; KLM	RC202-100	-	
RC103-25	332,8	KDMSS	RC203-25	356,4	KLM
RC103-50	710,9	KDMSS	RC203-50	-	
RC103-100	1258,0	CR; KLM	RC203-100	-	
RC104-25	306,6	KDMSS	RC204-25	-	
RC104-50	545,8	KDMSS	RC204-50	-	
RC104-100	-		RC204-100	-	
RC105-25	411,3	KDMSS	RC205-25	338,0	L; KLM
RC105-50	855,3	KDMSS	RC205-50	631,0	KLM
RC105-100	1513,7	KDMSS	RC205-100	-	
RC106-25	345,5	KDMSS	RC206-25	324,0	KLM
RC106-50	723,2	KDMSS	RC206-50	-	
RC106-100	-		RC206-100	-	
RC107-25	298,3	KDMSS	RC207-25	298,3	KLM
RC107-50	642,7	KDMSS	RC207-50	-	
RC107-100	-		RC207-100	-	
RC108-25	294,5	KDMSS	RC208-25	-	
RC108-50	598,1	KDMSS	RC208-50	-	
RC108-100	-		RC208-100	-	

Other than exact approaches, some authors were able to achieve good near optimal solutions in short competition times by use of metaheuristics. Two of these studies belong to Rochat and Taillard (1995) and Taillard et. al. (1997). The heuristics of Homberger and Gehring (1999) were also competitive. Kilby et al. (1998) and Chiang and Russel (1997) generated particularly good results for a few problems with long distances. In addition, Cordeau et. al. (2000) produced some best solutions for a number of instances (Cordeau et. al., 2002).

The best solutions achieved by heuristic approaches can be seen in Table 2.7. The studies are referred in the Table as follows: Rochat and Taillard (1995) (RT); Taillard et.

al. (1997) (TBGGP); Homberger and Gehring (1999) (HG); Kilby et al. (1998) (KPS); Chiang and Russel (1997) (CRu); Cordeau et. al. (2000) (CLM) (Cordeau et. al., 2002).

Table 2.7 Best known solutions achieved by heuristic approaches.

Short Horizon Problems			**Long Horizon Problems**		
Problem No	**Solution Cost**	**Authors**	**Problem No**	**Solution Cost**	**Authors**
R101	1650,8	RT	R201	1252,37	HG
R102	1486,12	RT	R202	1197,66	CLM
R103	1292,85	HG	R203	942,64	HG
R104	982,01	RT	R204	849,62	CLM
R105	1377,11	RT	R205	998,72	KPS
R106	1252,03	RT	R206	912,97	RT
R107	1113,69	CLM	R207	914,39	CRu
R108	964,38	CLM	R208	731,23	HG
R109	1194,73	HG	R209	910,55	HG
R110	1125,04	CLM	R210	955,39	HG
R111	1099,46	HG	R211	910,09	HG
R112	1003,73	HG			
C101	828,9	RT	C201	591,6	RT
C102	828,9	RT	C202	591,6	RT
C103	828,1	RT	C203	591,2	RT
C104	824,8	RT	C204	590,6	RT
C105	828,9	RT	C205	588,9	RT
C106	828,9	RT	C206	588,5	RT
C107	828,9	RT	C207	588,3	RT
C108	828,9	RT	C208	588,3	RT
C109	828,9	RT			
RC101	1696,94	TBGGP	RC201	1406,94	CLM
RC102	1554,75	TBGGP	RC202	1389,57	HG
RC103	1262,02	RT	RC203	1060,45	HG
RC104	1135,48	CLM	RC204	799,12	HG
RC105	1637,15	HG	RC205	1302,42	HG
RC106	1427,13	CLM	RC206	1156,26	KPS
RC107	1230,54	TBGGP	RC207	1062,05	CLM
RC108	1139,82	TBGGP	RC208	832,36	CLM

As artificial intelligence techniques became popular, many researchers have applied them to VRPTW. One of these studies is Garcia et. al.(1994). The authors have applied TS heuristic in parallel, which runs several neighborhoods of a solution on parallel processors at the same time. Badeau et. al.(1997) have proposed another parallel TS algorithm for VRPTW. They have shown that parallel application does not decrease solution quality while increasing solution speed.

In 1999, Liu and Shen developed a two stage metaheuristic algorithm. They have handled neighborhood construction in two stages. Valuable information is extracted from first stage of parallel neighborhood construction and used to build higher quality neighborhoods in the second stage. Tests of the algorithm have brought out competitive results.

Tan et. al. (2001) have applied several artificial intelligence techniques to the 56 test problems of Solomon. Among these techniques appear some well known local search methods, a hybrid heuristic of simulated annealing and Tabu Search (developed by the authors) and genetic algorithms. The results yielded competitive solutions for 23 of the test problems.

Lau et al. (2003) have studied VRPTW where exists a limited number of vehicles. They also allowed time windows to be relaxed by introducing a penalty cost for lateness. The authors have employed TS for this problem and tested it on the benchmark problems in the literature.

Another study of VRPTW belongs to Berger and Barkaoui (2004). They have proposed a parallel hybrid genetic algorithm for the problem. The algorithm evolves two populations simultaneously where one of them aims to minimize total distance traveled and the other minimizes time window violations to get a feasible solution. Computational results on benchmark instances showed that the proposed algorithm provides some new best known solutions.

Braysy et. al.(2004) developed an approach based on multi start local search (MSLS) and several new improvement heuristics. The authors have also introduced a new speed up technique and post optimized the solutions found by MSLS. They tested the approach on quite a large problem space.

In 2005, Zhong and Cole included backhauls into VRPTW. They considered the two cases where all linehauls are performed before backhauls and where linehauls and backhauls are performed simultaneously. The proposed approach first constructs an initial feasible solution then improves it through guided local search. Mester and Braysy (2005) have also employed guided local search. But they hybridized guided local search with evolution strategies in an iterative two stage procedure. The authors carried out a local search in the first stage and they formed the neighborhood by evolution strategies in the second stage. The algorithm is tested both on CVRP and VRPTW.

Homberger and Gehring (2005) have proposed a two phase hybrid metaheuristic. In the first phase, they try to minimize number of vehicles and in the second phase, the objective is to minimize total distance traveled by TS. In the study of Bouthillier and Crainic (2005), different evolutionary algorithms and TS works in parallel and the solutions achieved are collected in a solution warehouse. The exchanges are performed in this warehouse then parallel processes rework independently. By this procedure the authors were able to achieve the better solution of either the evolutionary algorithms or TS.

Russell and Chiang (2006) have used scatter search framework for VRPTW. Scatter search is a metaheuristic approach that combines solutions from a reference set to achieve improved solutions. In this study, the authors have developed a robust method which benefits from a set covering model to combine vehicle routes. Then a reactive TS and TS with a recovering feature is used to improve solutions. Alvarenga et al. (2007) have proposed a hybrid heuristic by employing GA and set partitioning together.

Hashimoto et al. (2006) allow time windows and traveling time constraints to be flexible by treating both of them as cost functions. Then the authors determine the routes of vehicles by local search. The latter problem is to determine the optimal start times of services. Finally, this problem is addressed by dynamic programming in the paper.

More recently, researchers started to incorporate other real life assumptions into VRPTW. One of these studies belong to Bent and Hentenryck (2006). The authors have handled the pick up and delivery problem with time windows. The paper proposes a hybrid two stage algorithm in which simulated annealing is used in the first stage to decrease number of routes and large neighborhood search is employed to decrease total traveling cost in the second stage. Fabri and Recht (2006) has also proposed an efficient heuristic for the pickup and delivery problem with time windows. The pick up and delivery problem with time windows has also been addressed by Doerner et. al. (2008). The authors have also incorporated the assumption that time windows are dependent on pick up times. The paper proposes several constructive heuristics and a branch and bound algorithm applied for the Austrian Red Cross logistics problem.

Azi et. al. (2007) also handled VRPTW but where a single vehicle performs several tours to serve all customers. The paper proposes a shortest path algorithm with resource constraints. Kim et. al. (2006) considers a problem similar to Azi et. al.(2007). They handle the waste collection problem with time windows where each vehicle performs more than one tour. A capacitated clustering based waste collection algorithm is developed for the solution where the aim is to minimize total traveling time, number of vehicles as well as balancing workload.

Hsu et. al. (2007) inserted randomness of food delivery process into VRPTW and studied the stochastic version. The authors considered not only the transportation and fixed costs but also the inventory and penalty costs of violating time windows. In addition to these studies, Dondo and Cerda (2007) studied the VRPTW with heterogeneous fleet and multiple depots. In the paper, a three phase heuristic is introduced where the problem is split into clusters in the first phase. Vehicles are assigned to clusters by mixed integer linear programming. Finally, vehicle arrival times to customer nodes are scheduled by also mixed integer linear programming.

2.5 Real Life Applications of VRP

As stated before, most of the studies listed in CVRP, HVRP, SDVRP and VRPTW sections are based on theory and test problems in the literature. In other words they appear in Region I of Figure 2.1. However, there also exist many case studies in the literature, which deal with real life applications. All these applications contain characteristics of the basic VRP (such as vehicle numbers and capacities, demand satisfaction etc.). But each of them may have different side constraints (such as time windows, periodic deliveries, multi depots, split deliveries etc.). That is, they appear in Region II of Figure 2.1. Some of the case studies and their application areas are given in Table 2.8 below.

Table 2.8 Some Vehicle Routing Applications

Application	**Location**	**Researhers**	**Year**
Newspaper distribution	Australia	Holt and Watts (Golden et. al., 2002)	1988
Food distribution	Alberta Canada	Erkut and McLean (Golden et. al., 2002)	1992
Waste collection	Hanoi Vietnam	Tung and Pinnoi	2000
Daily milk distribution	Athens Greece	Tarantilis and Kiranoudis	2001
Collection of recycling papers	Almada Portugal	Baptista et. al.	2002
Fresh meat distribution	Athens Greece	Tarantilis and Kiranoudis	2002
Network planning of parcel services	Austria	Wasner and Zapfel	2004
Routing of urban bus lines	Sao Paulo Brazil	Rodrigues et. al.	2006
Mail distribution	Australia	Hollis et. al.	2005
Fresh goods distribution	Izmir, Turkey	Mizrak Ozfirat, Ozkarahan	2007

CHAPTER THREE
OVERVIEW OF THE TOOLS EMPLOYED IN THE PROPOSED APPROACH

The main challenge in all types of vehicle routing problems (VRP) lies in the subtour elimination constraints. If subtours are not considered, the problem can be solved through mathematical modeling to optimum. However, insertion of subtour constraints into a mathematical model inflates the model very much and makes the problem np-hard.

Integer programming (IP) models cannot handle these constraints and the model does not start the solution procedure at all. We cannot achieve even an initial solution since IP offers us only the optimum when possible. However, by constraint programming (CP) not only the optimum solution but all solutions (or at least one solution) satisfying the constraints can be achieved. So, using CP instead of IP may provide more alternative ways to find a good or optimum solution to VRP. Therefore, in this dissertation study CP is employed to find the routes of vehicles.

Application of CP in VRP problems is very limited. One of the earlier studies in this area belong to Shaw (1998). The author proposed a local search method called large neighborhood search. This method explores a large neighborhood by removing some of the edges from the graph and reinserting these using a constraint based tree search. An extension of large neighborhood search is presented by Pisinger and Ropke (2007) by adding an adaptive layer to the method. The new algorithm called adaptive large neighborhood search is employed to solve several different versions of VRP. Aminu and Eglese (2006) have considered the Chinese postman problem with time windows. They have transformed the problem into an equivalent VRPTW and developed a constraint programming model. The results indicated that when the time windows are tight, optimal solutions can be achieved.

To the best of our knowledge, these studies are the only ones, which employ constraint programming in VRP. The results found in all three studies are encouraging to apply constraint programming technology to VRP.

In HVRP, there is one more point that makes the problem even more challenging. That is as the vehicles with different capacities and fixed costs are included in the problem, a tradeoff between total fixed costs and total traveling costs arises.

That is, as the capacity of a vehicle increases, the fixed cost also increases. If we try to minimize total fixed cost in the assignment of vehicles to subproblems, smaller vehicles in capacity would be selected. Hence the number of vehicles would increase (total number of tours visiting the depot increases) and total distance traveled would increase accordingly.

On the other hand, if the objective function is minimizing total number of vehicles (instead of fixed cost) in the vehicle assignment, then the algorithm would tend to select higher capacity but more expensive vehicles. In this case total traveling cost would decrease but total fixed cost would increase. The tradeoff between fixed cost and traveling cost can be seen in Figure 3.1.

When there exist multiple objectives in a problem and the degree of fulfillment of these objectives is vague, fuzzy mathematical programming may be a useful tool to handle the problem (Zimmerman, 1978). Therefore in the proposed approach, interactive fuzzy goal programming (IFGP) is employed to assign vehicles to nodes.

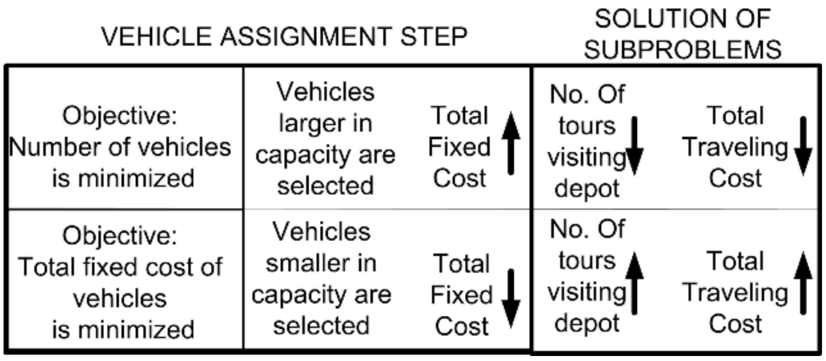

VEHICLE ASSIGNMENT STEP
SOLUTION OF SUBPROBLEMS
Objective: Number of vehicles is minimized
Vehicles larger in capacity are selected
Total Fixed Cost
No. Of tours visiting depot
Total Traveling Cost
Objective: Total fixed cost of vehicles is minimized
Vehicles smaller in capacity are selected
Total Fixed Cost
No. Of tours visiting depot
Total Traveling Cost

Figure 3.1 Tradeoff between total fixed cost and total traveling cost

Very few studies exist in literature, which consider fuzzy methods for VRPs. These studies can be grouped into two. In the first group, there exists a fuzzy component in the problem. Therefore, fuzzy methods are employed for the solution. For example, Teodorovic and Pavkovic (1996) considered CVRP where the demand at nodes are fuzzy. They proposed a heuristic based on the Sweep algorithm and fuzzy logic. Another study in this group belongs to Zheng and Liu (2006). The authors handled VRPTW with fuzzy traveling times and developed a fuzzy optimization model for the problem.

In the second group of fuzzy studies, authors employ fuzzy clustering in order to cluster the customer nodes and achieve solvable size VRPs. Some of the studies in this group are Hu and Sheu (2003), Sheu (2007) and Saez et. al. (2008). Hu and Sheu (2003) develops a fuzzy clustering method based on customer demand rather than geographical locations. Sheu (2007) employs the same approach and in addition uses multiobjective programming to solve the clustered problems. Saez et. al (2008), on the other hand, develops a fuzzy clustering method and employs it to solve the dynamic pick up and delivery problem. As can be seen from these studies, there is a lot more to explore in the use of fuzzy methods for VRPs.

In this chapter, CP and IFGP is explained respectively so that one can understand the advantages of these tools and how they will be applied in the proposed study in this dissertation.

3.1 Constraint Programming

In the last few years, CP has attracted high attention among experts from many areas because of its potential for solving hard real life problems. Not only it is based on a strong theoretical foundation but it is attracting widespread commercial interest as well, in particular, in areas of modeling heterogeneous optimization and satisfaction problems.

Constraints arise in most areas of human endeavor. They formalize the dependencies in physical worlds and their mathematical abstractions naturally and transparently. A constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain. The constraint thus restricts the possible values that variables can take, it represents partial information about the variables of interest. Constraints can also be heterogeneous, so they can bind unknowns from different domains, for example the length (number) with the word (string). Constraints naturally have several properties:

- Constraints may specify partial information, i.e., the constraint need not uniquely specify the values of its variables, (constraint X>2 does not specify the exact value of variable X, so X can be equal to 3, 4, 5 etc.)
- Constraints are heterogeneous, i.e., they can specify the relation between variables with different domains (for example X = length(Y))
- Constraints are non-directional, typically a constraint on two variables X, Y can be used to infer a constraint on X given a constraint on Y and vice versa, (X=Y+2 can be used to compute the variable X using X:=Y+2 as well as the variable Y using Y:=X-2)
- Constraints are declarative, i.e., they specify what relationship must hold without specifying a computational procedure to enforce that relationship,
- Constraints are additive, i.e., the order of imposition of constraints does not matter, all that matters at the end is that the union of constraints.
- Constraints are rarely independent, typically constraints in the constraint store share variables.

- Constraints can be stated either implicitly, e.g., an arithmetic formula, or explicitly, where each constraint is expressed as a set of values that satisfy the constraint. An example of an implicitly stated constraint on the integer variables x and y is $x < y$. An example of an explicitly stated constraint on the integer variables x and y with domains {1, 2, 3} and {1, 2, 3, 4} is the set {(1, 1), (2, 3), (3, 4)}.

CP is the study of computational systems based on constraints. The idea of constraint programming is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding solution satisfying all the constraints.

CP aims to solve combinatorial optimization problems. Often these combinatorial optimization problems are solved by defining them as one or several instances of the Constraint Satisfaction Problem (CSP). An instance of a CSP is described by a set of variables, a set of possible values for each variable, and a set of constraints between the variables. The set of possible values of a variable is called the variable's domain. A constraint between variables expresses which combinations of values for the variables are allowed. The question to be answered for an instance of the CSP is whether there exists an assignment of values to variables, such that all constraints are satisfied. Such an assignment is called a solution of the CSP.

In short, a CSP consists of:

- a set of variables $\boldsymbol{X}=\{\boldsymbol{x_1},...,\boldsymbol{x_n}\}$,
- for each variable $\boldsymbol{x_i}$, a finite set $\boldsymbol{D_i}$ of possible values (its domain),
- and a set of constraints restricting the values that the variables can simultaneously take.

A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied. The objective may be to find:

- just one solution, with no preference as to which one,

• all solutions,

• an optimal, or at least a good solution, given some objective function defined in terms of some or all of the variables; in this case the problem is called Constraint Optimization Problem (COP).

3.1.1 Basic Problems of CSP

Solutions to a CSP can be found by searching systematically through the possible assignments of values to variable. Search methods divide into two broad classes, those that traverse the space of partial solutions (or partial value assignments), and those that explore the space of complete value assignments (to all variables) stochastically.

Some of the basic problems of CSP are the graph (map) colouring, N-queens, and crypto-arithmetic problems. These problems reflect the typical characteristics of a CSP so that one can understand the logic of CP.

• N-Queens: The N-queens problem is a well know puzzle among computer scientists. Given any integer ***N***, the problem is to place ***N*** queens on squares in an ***N*N*** chessboard satisfying the constraint that no two queens threaten each other (a queen threatens any other queens on the same row, column and diagonal).

One possible representation of this problem is by using ***n*** decision variables as $x_1,..,x_n$, each with domain [1..n]. The idea is that x_i denotes the row number that the queen on column ***i*** stands. The appropriate constraints to state that no two queens threaten each other are:

- $x_i \neq x_j$ (no two queens in the same row);
- $x_i - x_j \neq i-j$ (no two queens in each South-West – North-East diagonal);
- $x_i - x_j \neq j-i$ (no two queens in each North-West – South-East diagonal);

where $i \in [1..n-1]$ and $j \in [i+1..n]$. (Apt, 2003).

- Graph (map) colouring: Another problem, which is often used to demonstrate potential areas of CP and to explain concepts and algorithms for the CSP is the colouring problem. Given a graph (a map) and a number of colours, the problem is to assign colours to those nodes (areas in the map) satisfying the constraint that no adjacent nodes (areas) have the same colour assigned to them.

This problem is modeled naturally by labeling each node of the graph with a variable (with the domain corresponding to the set of colours) and introducing the non-equality constraint between each two variables labeling adjacent nodes. An example of this problem can be seen in Figure 3.2.

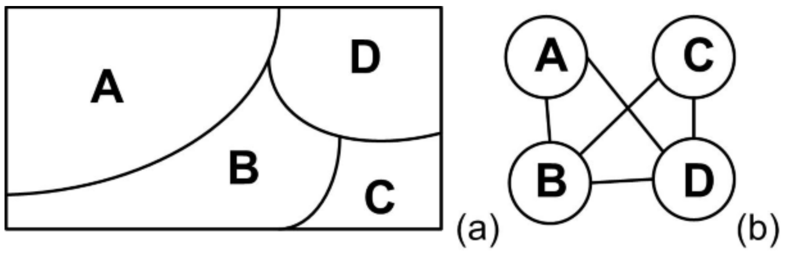


Figure 3.2 (a)A variable is associated with each of the A, B, C and D regions.
(b)The lines connecting adjacent nodes are inequality constraints.

- Crypto-arithmetic: Given a mathematical expression where letters are used instead of numbers, the problem is to assign digits to those letters satisfying the constraint that different letters should have different digits assigned and the mathematical formulae holds. Here is a typical example of the crypto-arithmetic problem:

$$\text{SEND} + \text{MORE} = \text{MONEY} \qquad (3.1)$$

Here, the problem can be modeled by identifying each letter with a variable. Since S and M are leading digits, their domain is [1..9]. Domain of all other variables is [0..9]. The constraints of the problem are:

$$\begin{array}{rrrrrrrrr} & 1000S & + & 100E & + & 10N & + & D \\ + & 1000M & + & 100O & + & 10R & + & E \\ = 10000M & + 1000O & + & 100N & + & 10E & + & Y \end{array} \tag{3.2}$$
$$x \neq y, x \in \{S,E,N,D,M,O,R,Y\}, y \in \{S,E,N,D,M,O,R,Y\}$$

(Apt, 2003).

3.1.2 Concepts to Solve CSP

In general, the tasks posed in the CSP paradigm are np-hard. Some of these tasks are listed in the following:

- **Consistency techniques:** In CP**,** various consistency techniques were introduced to prune the search space. The consistency techniques are node-consistency, arc-consistency and path consistency.
- **Constraint propagation:** The process of actively using constraints to come to certain deductions is called constraint propagation.
- **Systematic search algorithm:** A CSP can be solved using *generate-and-test paradigm* (GT) that systematically generates each possible value assignment and then it tests to see if it satisfies all the constraints. A more efficient method uses the *backtracking paradigm* (BT) that is the most common algorithm for performing systematic search. Backtracking incrementally attempts to extend a partial solution toward a complete solution, by repeatedly choosing a value for another variable.
- **Constraint optimization:** A typical real-life problem is not only about finding a solution satisfying all the constraints. Frequently, some optimization is involved and the customers are asking for good solutions. The optimization nature of the problem can be integrated by an objective function defined in terms of problem variables. Many constraint satisfaction algorithms can be extended to solve optimization problems.

3.1.2.1 Consistency Techniques

The first approach to solving CSP is based on removing inconsistent values from variables' domains until the solution is achieved. These methods are called consistency techniques and they were introduced first in the scene labeling problem (Bartak, 1999). Mainly there are three types of consistency techniques. Node consistency is the first one which deals with unary constraints. Arc consistency deals with binary constraints and finally path consistency deals with more than one constraint at a time (Apt, 2003).

- **Node Consistency:** Node consistency is the most common notion of consistency. It deals with unary constraints.

Definition: A CSP is called node consistent if for every variable ***x*** every unary constraint on ***x*** coincides with the domain of ***x***.

Example: Consider a CSP of the form

$$\{C, x_1 \geq 0, \ldots\ldots, x_n \geq 0; x_1 \in N, \ldots\ldots, x_n \in N\} \quad (3.3)$$

where ***C*** does not contain unary constraints and ***N*** denotes the set of natural numbers. This CSP is node consistent since for each variable, its unary constraint is satisfied by all the values in the variable's domain.

Example: Consider a CSP of the form

$$\{C, x_1 \geq 0, \ldots\ldots, x_n \geq 0; x_1 \in N, \ldots\ldots, x_{n-1} \in N, x_n \in Z\} \quad (3.4)$$

where ***C*** does not contain unary constraints and ***Z*** denotes the set of all integers. This CSP is not node consistent since for the variable $\boldsymbol{x_n}$, the constraint $\boldsymbol{x_n}\geq 0$ is not satisfied by the negative integers in its domain.

- **Arc Consistency:** Arc consistency deals with binary constraints. Basically, it can be said that a binary constraint is arc consistent if every value in each domain participates in the solution. In addition, a CSP is said to be arc consistent if all its binary constraints are arc consistent.

Definition 2: Given a constraint ***C*** over ***n*** variables $x_1,\ldots, x_n$ and a domain $d(x_i)$ for each variable x_i, ***C*** is said to be arc-consistent if and only if for any variable x_i and any value v_i in $d(x_i)$, there exist values $v_1,\ldots, v_{i-1}, v_{i+1},\ldots, v_n$ in $d(x_1),\ldots, d(x_{i-1}), d(x_{i+1}),\ldots, d(x_n)$ such that $C(v_1,\ldots, v_n)$ holds.

Example: Consider the CSP which consists of only one constraint, $x<y$, where domain of ***x***, D_x=[5..10] and domain of ***y***, D_y =[11..15]. This CSP is arc consistent since for every value in domain of ***x***, there exists a value in domain of ***y*** which satisfies the constraint.

Example: Consider the same CSP which consists of only one constraint, $x<y$, where domain of ***x***, D_x=[5..10] and domain of ***y***, D_y =[3..7]. This CSP is not arc consistent. For instance, consider the value 8 in domain of ***x***. In domain of ***y***, there is no value ***b*** that satisfies $8<b$.

- **Path consistency:** The first two methods of inconsistency deal with constraints on their own. However using only these two may be very time consuming. For example consider the following CSP:

$$\{x \prec y, y \prec z, z \prec x, x \in \{1..100000\}, y \in \{1..100000\}, z \in \{1..100000\}\} \tag{3.5}$$

If we are trying to prove that this CSP is inconsistent, we can use arc consistency. First, the constraint, x<y is considered. Then the CSP becomes:

$$\{x \prec y, y \prec z, z \prec x, x \in \{1..99999\}, y \in \{1..100000\}, z \in \{1..100000\}\} \tag{3.6}$$

Next, the constraint, z<x is considered. Then the CSP becomes:

$$\{x \prec y, y \prec z, z \prec x, x \in \{1..99999\}, y \in \{1..100000\}, z \in \{1..99998\}\} \tag{3.7}$$

Similarly, the process will go on iteratively until the CSP fails. But, to understand that this CSP is inconsistent would take a huge number of steps by arc consistency. Therefore, path consistency is introduced which deals with two constraints at a time. In order to understand path consistency, three notions will be introduced first.

Definition:

- A CSP is called normalized if for each pair ***x***, ***y*** of its variables at most one constraint on ***x*** and ***y*** exists in the CSP.
  Given a normalized CSP and a pair ***x***, ***y*** of its variables, $\boldsymbol{C_{x,y}}$ denotes, the constraint on ***x*** and ***y*** if it exists. Otherwise, it represents the relation on ***x*** and ***y*** that equals the Cartesian product of the domains of the variables ***x*** and ***y***.
- A CSP is called standardized if for each pair ***x***, ***y*** of its variables, a unique constraint on ***x*** and ***y*** exists in the CSP.
- A CSP is called regular if for each sequence ***X*** of its variables, a unique constraint on ***X*** exists in the CSP. $\boldsymbol{C_X}$ denotes the unique constraint on ***X***.

Definition: A normalized CSP is path consistent if for each subset {x, y, z} of its variables,

$$C_{x,z} \subseteq C_{x,y} \times C_{y,z} \qquad (3.8)$$

holds.

In other words, a normalized CSP is path consistent if for each subset {x, y, z} of its variables the following holds:

If (***a***, ***c***) Є $\boldsymbol{C_{x,z}}$, then there exists ***b*** such that (***a***, ***b***) Є $\boldsymbol{C_{x,y}}$ and (***b***,***c***) Є $\boldsymbol{C_{y,z}}$.

Example: Consider the following normalized CSP,

$$\{x \prec y, y \prec z, x \prec z, x \in [0..4], y \in [1..5], z \in [6..10]\} \quad (3.9)$$

The graphical representation of the CSP can be seen in Figure 3.3.

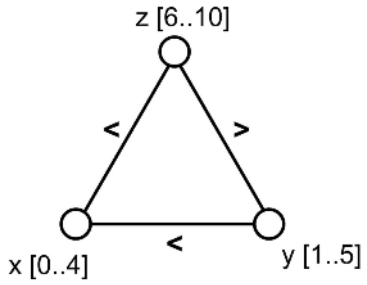


Figure 3.3 A path consistent CSP.

This CSP is path consistent since the following three relations hold:

$$\begin{aligned} C_{x,y} &= \{(a,b) \mid a \prec b, a \in [0..4], b \in [1..5]\} \\ C_{x,z} &= \{(a,c) \mid a \prec c, a \in [0..4], c \in [6..10]\} \\ C_{y,z} &= \{(b,c) \mid b \prec c, b \in [1..5], c \in [6..10]\} \end{aligned} \quad (3.10)$$

Example: Consider now the following normalized CSP,

$$\{x \prec y, y \prec z, x \prec z, x \in [0..4], y \in [1..5], z \in [5..10]\} \quad (3.11)$$

This CSP is not path consistent since the following relation does not hold:

$$C_{y,z} = \{(b,c) \mid b \prec c, b \in [1..5], c \in [5..10]\} \quad (3.12)$$

All above mentioned consistency techniques are covered by a general notion of ***K***-consistency and strong ***K***-consistency. A constraint graph is ***K***-consistent if for every system of values for ***K***-1 variables satisfying all the constraints among these variables, there exits a value for arbitrary ***K***th variable such that the constraints among all ***K*** variables are satisfied. A constraint graph is strongly ***K***-consistent if it is ***J***-consistent for all ***J***Є***K***. It should be noted that

- node consistency is equivalent to strong 1-consistency,
- arc consistency is equivalent to strong 2-consistency,
- path consistency is equivalent to strong 3-consistency.

Clearly, if a constraint graph containing ***N*** nodes is strongly ***N***-consistent, then a solution to the CSP can be found without any search. But the worst case complexity of the algorithm for obtaining ***N***-consistency in an ***N***-node constraint graph is exponential (Bartak, 1999).

Algorithms exist for making a constraint graph strongly K-consistent for K>2 but in practice they are rarely used because of efficiency issues. Although these algorithms remove more inconsistent values than any arc-consistency algorithm they do not eliminate the need for search in general.

3.1.2.2 Constraint Propagation

In general, CSPs can be solved by enumeration, but this approach is not practical since the search space grows exponentially with the dimension of the problem. One of the key ideas of CP is that constraints can be used actively to reduce the computational effort needed to solve combinatorial problems. This process of actively using constraints to come to certain deductions is called constraint propagation. Thus, CSPs greatly benefit from the reduction of the search space performed by constraint propagation.

Example: $y > x$

$x > 8$ (3.13)

$y < 9$

x and y integers

Looking at the first two constraints, it can be said that the value of y is greater than 9. Then by using the constraint $y < 9$, a contradiction is detected.

The algorithms that achieve local consistency (node consistency, arc consistency, path consistency) are called constraint propagation algorithms. In the literature, several other names exist for constraint propagation algorithms such as local propagation, consistency enforcing, filtering and narrowing algorithms etc.

The propagation algorithm is part of the constraint itself, and it is triggered when an event takes place due to a domain modification of one of the variables. The modification in the domain can be one of the following:

- reduction of a domain bound,
- removal of a value,
- domain may be reduced to a single value (instantiation of a variable).

As soon as one constraint produces a modification on the domain of a variable $\boldsymbol{X}$, all constraints involving $\boldsymbol{X}$ are triggered to perform propagation according to the change in the domain of $\boldsymbol{X}$. During computation, constraints are propagated in order to reduce variable domains by removing inconsistent values. If a domain becomes empty then a failure is faced and backtracking takes place.

Example: One of the basic propagation algorithms is the one to achieve arc consistency. Consider the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ where $\boldsymbol{D_x}$=[1..10] and $\boldsymbol{D_y}$=[4..18] linked via the constraint $\boldsymbol{x>y}$. This constraint is not arc consistent. Values of 1, 2, 3 and 4 for variable $\boldsymbol{x}$ do not have a support in $\boldsymbol{y}$. Similarly, values 10 to 18 in $\boldsymbol{y}$ do not have a support in $\boldsymbol{x}$. Therefore these values should be removed from the domains in order to make the constraint arc consistent. New $\boldsymbol{D_x}$=[5..10] and new $\boldsymbol{D_y}$=[4..9].

3.1.2.3 Systematic Search

In the result of constraint propagation process, three alternative situations may arise:

- a domain becomes empty and failure occurs
- a solution is found
- some domains contain more than one value.

In the third case, constraint propagation is not complete. Consequently, one needs to perform some kind of search to determine if the CSP instance at hand has a solution or not. Most commonly, search is performed by means of a tree search algorithm. One widely used way for the search is called labeling which means that one variable is selected and one value in its domain is assigned to the variable itself. For example, consider the selected variable is $\boldsymbol{x_i}$ in problem $\boldsymbol{P}$ and its domain is $\boldsymbol{D_i}$ which contains the values $\boldsymbol{v_{1i}}, \ldots, \boldsymbol{v_{bi}}$. By labeling, the problem is partitioned into $\boldsymbol{b}$ subproblems where

$$\begin{aligned} SP_1 &= P \cup \{x_i = v_{1i}\} \\ SP_2 &= P \cup \{x_i = v_{2i}\} \\ &\ldots\ldots \\ SP_b &= P \cup \{x_i = v_{bi}\} \end{aligned} \qquad (3.14)$$

The constraints $\{\boldsymbol{x_i}=\boldsymbol{v_{ki}}\}$ are called branching constraints (Representation of a search tree can be seen in Figure 3.4). During the search, constraints are taken into account and propagated in order to prune the search space as much as possible. A variable instantiation triggers all constraints involving that variable and propagation process starts again.

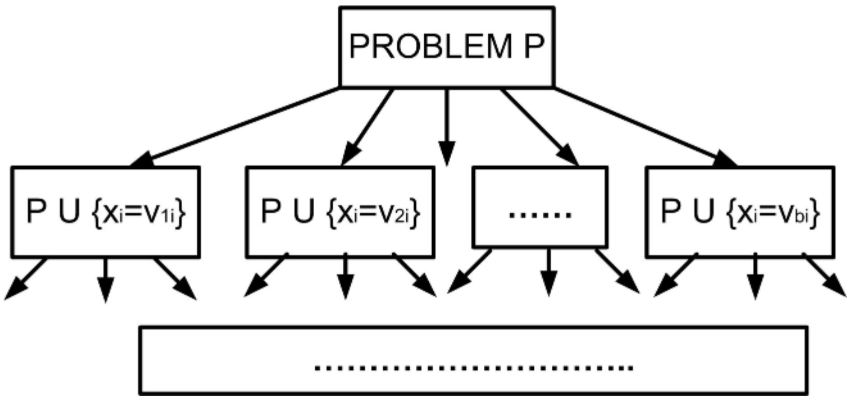


Figure 3.4 Representation of a search tree

The way the variables are selected and the order of its values greatly effects the performance of the tree search. Some of the well known search strategies are listed below (ILOG, 2003):

- Best First Search (BFS): This strategy maintains all unexplored nodes and goes on exploring the one with the best lower bound. It is a good strategy where there exist precise lower bounds.

- Slice Based Search (SBS): This is a strategy which assumes the existence of a good heuristic. It searches the solution space, allowing different decisions for the heuristic. The decisions allowed to change are called discrepancies. It is effective in problems where strong heuristics exist.
- Depth Bounded Discrepancy Search (DDS): This strategy is similar to SBS. In addition it favors the discrepancies which occur in the lower levels of the search tree.
- Depth First Search (DFS): This search strategy progresses by expanding the first child node of the search tree that appears and thus goes deeper and deeper until a goal node is found, or until a node that has no children appears. Then the search backtracks, returning to the most recent node it had not finished exploring.
- Interleaved Depth First Search (IDFS): This strategy simulates a parallel depth first search exploration.

The way search space is explored greatly influences the performance of the overall constraint based computation. All CP languages have high level backgrounds allowing to easily write search heuristics. This leads to the development of sophisticated branching methods for many type of problems allowing to solve them effectively.

3.1.2.4 Constraint Optimization

The quality of solution is usually measured by the objective function. The goal is to find such solution that satisfies all the constraints and minimize or maximize the objective function respectively. Such problems are referred as Constraint Optimization Problems (COP).

A Constraint Optimization Problem (COP) consists of a standard CSP and an optimization function that assigns every solution to a numerical value.

The most widely used algorithm for finding optimal solutions is Branch and Bound (B&B) and it can be applied to COP as well. The efficiency of B&B is determined by two factors: the quality of the heuristic function and whether a good bound is found early. Observations of real-life problems show also that the “last step” to optimum, i.e., improving a good solution even more, is usually the most computationally expensive part of the solving process. Fortunately, in many applications, users are satisfied with a solution that is close to optimum if this solution is found early. B&B algorithm can be used to find suboptimal solutions as well by using the second “acceptability” bound. If the algorithm finds a solution that is better than the acceptability bound then this solution can be returned to the user even if it is not proved to be optimal.

3.1.3 CP Working Procedure

By using constraint propagation, search procedures and inconsistency techniques, the way CP works is given in Figure 3.5. Once the problem is defined, the constraints and variables are turned into a partial solution. Then, in addition to these, CP tools offer ways to define new constraints using constraint propagation that can be used with the predefined constraints. Also, CP tools specify the search heuristic and backtracking strategy (or give the user to specify these). This procedure works iteratively until a full solution is achieved (or optimal solution in case of COP).

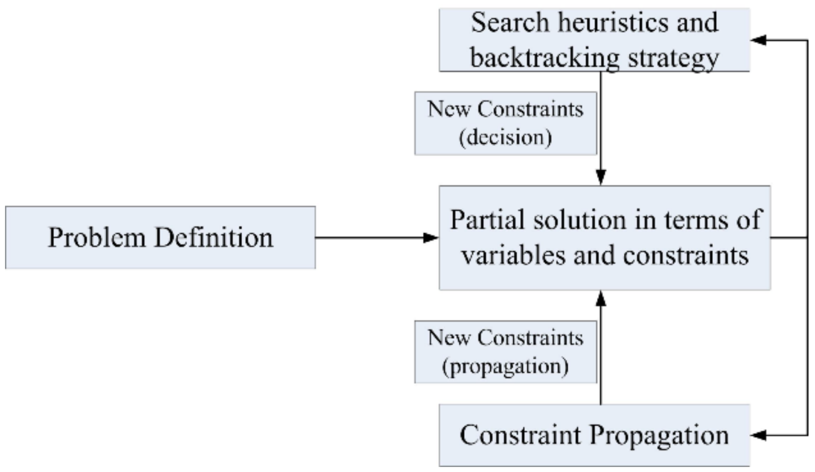

Search heuristics and backtracking strategy
New Constraints (decision)
Problem Definition
Partial solution in terms of variables and constraints
New Constraints (propagation)
Constraint Propagation

Figure 3.5 The behavior of CP system (Baptiste et. al., 2003)

One advantage of CP is the existence of global constraints. Global constraints are expressive and powerful constraints embedding constraint dependent filtering algorithms. A typical global constraint is the

$$\text{alldifferent}([\,\boldsymbol{X_1}, ..., \boldsymbol{X_n}]) \qquad (3.15)$$

available in most CP solvers. This constraint holds if and only if all variables are assigned a different value. Thus, it is declaratively equivalent to a set of (n*(n-1)/2) binary inequality constraints. An extension of the alldifferent is the global cardinality constraint:

$$\text{gcc}([\,\boldsymbol{X_1}, ..., \boldsymbol{X_n}], [\boldsymbol{v_1}, ..., \boldsymbol{v_m}], [\,\boldsymbol{l_1}, ..., \boldsymbol{l_m}], [\,\boldsymbol{u_1}, ..., \boldsymbol{u_m}]\,) \; (3.16)$$

which holds if and only if the number of variables in $[\boldsymbol{X_1}, ... ,\boldsymbol{X_n}]$ which assume value v_i is within $\boldsymbol{1_i}$ and $\boldsymbol{u_i}$. A further extension of the global cardinality constraint is the global sequencing constraint:

$$\text{gsc}([\,\boldsymbol{X_1}, ..., \boldsymbol{X_n}], [\boldsymbol{v_1}, ..., \boldsymbol{v_m}], \boldsymbol{min}, \boldsymbol{max}, \boldsymbol{q}, [\,\boldsymbol{l_1}, ..., \boldsymbol{l_m}], [\,\boldsymbol{u_1}, ..., \boldsymbol{u_m}]) \quad (3.17)$$

This constraint holds if and only if each $\boldsymbol{v_i}$ appears a number of times between $\boldsymbol{l_i}$ and $\boldsymbol{u_i}$ and for each sequence $\boldsymbol{S_i}$ of $\boldsymbol{q}$ consecutive variables, each $\boldsymbol{v_i}$ appears a number of times between ***min*** and ***max***. Other important constraints in CP have the form:

$$\begin{aligned}&\text{atmost}(\boldsymbol{i}, [\,\boldsymbol{X_1}, ..., \boldsymbol{X_n}], \boldsymbol{v})\\ &\text{atleast}(\text{i}, [\,\boldsymbol{X_1}, ..., \boldsymbol{X_n}], \boldsymbol{v}) \qquad (3.18)\\ &\text{exactly}(\text{i}, [\,\boldsymbol{X_1}, ..., \boldsymbol{X_n}], \boldsymbol{v})\end{aligned}$$

meaning respectively that at most, at least and exactly ***i*** variables in the list assume the value ***v***.

3.1.4 Applications

Constraints have recently emerged as a research area that combines researchers from a number of fields, including artificial intelligence, programming languages, symbolic computing and computational logic. Constraint networks and CSPs have been studied in artificial intelligence starting from the seventies. Due to its potential to model and solve real-life problems naturally and efficiently, CP has been successfully applied in numerous domains. CP can also serve as a roof for combination of different approaches, like integer programming and operation research.

There are two main reasons for choosing to represent and solve a problem as a CSP rather than a mathematical programming problem.

- First, the representation as a CSP is often much closer to the original problem: the variables of the CSP directly correspond to problem entities, and the constraints can be expressed without having to be translated into linear inequalities. This makes the formulation simpler, the solution easier to understand, and the choice of good heuristics to guide the solution strategy more straightforward.
- Second, although CSP algorithms are essentially very simple, they can sometimes find solution more quickly than integer programming methods.

Recent applications include computer graphics, natural language processing, database systems, operations research problems, molecular biology, business applications, electrical engineering, circuit design, etc. It proves itself to be well adapted to solving real life problems because many application domains involve constraints naturally. Current research in this area deals with various foundational issues, with implementation aspects, and with new applications of CP.

There are a lot of companies providing solutions based on constraints like PeopleSoft, i2 Technologies, InSol, Vine Solutions or companies providing constraint-based tools like ILOG, IF Computer, Cosytec, SICS, or PrologIA.

The constraint programming techniques can be used in many real-life problems. Among these are:

- time-tabling
- workforce management
- course scheduling
- stuff scheduling
- nurse scheduling
- crew rostering problem
- planning and scheduling
- transport planning
- on-demand manufacturing
- car sequencing
- resource allocation
- forest treatment scheduling
- well activity scheduling
- airport counter allocation
- analysis and synthesis of analogue and digital circuits
- option trading analysis
- cutting stock
- DNA sequencing
- chemical hypothetical reasoning
- warehouse location
- network configuration

Assignment problems were perhaps the first type of industrial application that were solved with the constraint tools. A typical example is the stand allocation for airports, where aircraft must be parked on the available stand during the stay at airport (Dincbas, Simonis, 1991) or counter allocation for departure halls (Chow and Perett, 1997). Another example is berth allocation to ships in the harbor (Perett, 1991).

Another typical constraint application area is personnel assignment where work rules and regulations impose difficult constraints. The important aspect in these problems is the requirement to balance work among different persons. Applications in this area consist of rosters for nurses in hospitals, crew assignment to flights or staff assignment in railways companies (Focacci et. al. 1997).

Probably the most successful application area for finite domain constraints is scheduling problems, where, again, constraints express naturally the real life limitations. The usage of constraints in advanced planning and scheduling systems increases due to current trends of on-demand manufacturing.

Extensive application usage of CP in solving real-life problems includes a number of limitations and shortcomings.

As many problems solved by CP belong to the area of NP-hard problems, the identification of restrictions that make the problem tractable is very important both from the theoretical and the practical points of view. However, as with most approaches to NP-hard problems, efficiency of constraint programs is still unpredictable and the intuition is usually the most important part of decision when and how to use constraints. The most common problem stated by the users of the constraint systems is stability of the constraint model. Even small changes in a program or in the data can lead to a dramatic change in performance.

Another problem is choosing the right constraint satisfaction technique for particular problem. In addition, a particular problem in many constraint models is the cost optimization. Sometimes, it is very difficult to improve an initial solution, and a small improvement takes much more time than finding the initial solution. There is a trade off between any solution and best solution.

3.1.5 Future Research Trends

The shortcomings of current constraint satisfaction systems mark the directions for the future development. Among them, modeling looks one of the most important. The discussions started about using of global constraints that encapsulate primitive constraints into a more efficient package (e.g., alldifferent constraint). A more general question concerns modeling languages to express constraint problems. Currently, most CP packages are either extensions of a programming language or libraries that are used with conventional programming languages. Some of the well known CP tools are ILOG SOLVER (Puget, 1994; Puget and Leconte, 1995), CHIP (Hentenryck, 1999; Adhikary et. al. 1997; Johansen, 1997), PROLOG III, IV (Colmerauer, 1990), ECLiPSe (Wallace et. al., 1997), CLAIRE (Caseau and Laburthe, 1996) and Choco (Laburthe, 2000) (Baptiste et. al., 2003).

The study of interactions of various constraint solving methods is one of the most challenging problems. The hybrid algorithms combining various constraint solving techniques are under study by many researchers (Jacquet-Lagreze, 1998). The combination of constraint satisfaction techniques with traditional OR methods like integer programming is another challenge of current research. Finally, parallelism and concurrent constraint solving (Hentenryck, 1992) are studied as methods for improving efficiency.

3.2 Interactive Fuzzy Goal Programming

The main objective of this section is to review the basic concepts of IFGP which is employed in the proposed methodologies in this research. In order to understand the basics of IFGP, firstly, fuzzy set theory (FST) proposed by Zadeh (1965), is introduced briefly. Then, fuzzy linear programming (FLP) and fuzzy goal programming (FGP) are explained so that the logic and working procedure of IFGP can be understood.

3.2.1 Fuzzy sets

Fuzzy sets are a generalization of conventional set theory that was introduced by Zadeh in 1965 as a mathematical way to represent vagueness in everyday life (Bezdek, 1993). Since then, a huge number of fuzzy methods have been developed by the researchers who study operations research and artificial intelligence, and quite a number real-world problems have been successfully solved using fuzzy methods.

In real life, some information can only be approximately determined. For instance, "*The temperature is about 37°C*" shows that one value around 37 is true but not known exactly. This situation can be defined by an ordinary set in which the set of numbers L from 36 to 38 is crisp, and, can be written as; $L = \{r \in \Re \mid 36 \le r \le 38\}$. And also, the *characteristic function* of this set is as follows (Bezdek, 1993):

$$C_L(r) = \begin{cases} 1 & 36 \le r \le 38 \\ 0 & otherwise \end{cases} \text{ and } C_L : \Re \to \{0,1\} \tag{3.19}$$

The values of C_L is equal to 1, when r is in L; otherwise C_L is equal to zero. So ordinary sets correspond to two–valued logic: 1 or 0 (Bezdek, 1993).

Unlike the ordinary set, this situation can be defined by a fuzzy set using the membership function concept. The membership function of a fuzzy set has values between 0 and 1, which denote the degree of membership of a member in the given set.

In general, a fuzzy set is defined as follows (Sakawa, 1993):

"Let X denotes a universal set. Then a fuzzy set F in X is defined as a set of ordered pairs $F = \{(x, \mu_F(x)) \mid x \in X\}$, where, $\mu_F(x)$ is called the membership function for the fuzzy set F. The $\mu_F(x)$ represents the grade of membership of x in F. Thus, the nearer the value of $\mu_F(x)$ is unity, the higher the grade of membership of x in F."

When X is a finite set whose elements are $x_1, x_2, ..., x_n$, a fuzzy set *F* on *X* is expresses as (Sakawa, 1993):

$$F = \{(x_1, \mu_F(x_1)), (x_2, \mu_F(x_2), ..., (x_n, \mu_F(x_n))\} \qquad (3.20)$$

In literature, there are very different ways to state fuzzy sets. For example, Zadeh (1965) states this fuzzy set as:

$$F = \mu_F(x_1)/x_1 + \mu_F(x_2)/x_2 +,....,+\mu_F(x_n)/x_n = \sum_{i=1}^{n} \mu_F(x_i)/x_i \qquad (3.21)$$

Before defining the basic operations in FST, basic definitions about fuzzy sets should be given as follows (Zimmerman, 1996):

- "The support of a fuzzy set F, S(F), is the crisp set of all x∈X such that $\mu_F(x) > 0$.
- A fuzzy set with a membership function that has a grade of 1 is called normal. In other words, A is called "normal" $\leftrightarrow \max_{x \in X} \mu_F(x) = 1$.
- A fuzzy set F is convex if
  $\mu_F(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu_F(x_1), \mu_F(x_2)\}, x_1, x_2 \in X, \lambda \in [0,1]$
- The crisp set of elements that belong to the fuzzy set F at least to the degree α is called the α-level set:
  $F_\alpha = \{x \in X \mid \mu_F(x) \geq \alpha\}$
  $F_\alpha = \{x \in X \mid \mu_F(x) > \alpha\}$ is called strong α-level set or strong α-cut".

3.2.1.1 Basic operations in fuzzy set theory

Union, intersection, and complement are the basic operations in classical set theory. Fuzzy sets have also similar operations; however, these operations are defined using the membership functions as follows (Sakawa, 1993; Zimmerman, 1996):

- Intersection: The intersection of two fuzzy sets A and B is defined by the membership function $\mu_C(x)$ of the intersection $C = A \cap B$ as follows:

$\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}, \quad x \in X$,

- Union: The union of two fuzzy sets A and B is defined by the membership function $\mu_D(x)$ of the union $D = A \cup B$ as follows:

  $\mu_D(x) = \max\{\mu_A(x), \mu_B(x)\}, \quad x \in X$,

- Complementation: The membership function of the complement of a normalized fuzzy set A, denoted by $\overline{A}$, is defined as follows:

  $\mu_{\overline{A}}(x) = 1 - \mu_A(x), \qquad x \in X$.

3.2.2 Fuzzy numbers

For a normal and convex fuzzy set, if a weak α-cut (α level-set) is a closed interval, it is called a fuzzy number (Terano et al., 1992). Fuzzy numbers are used to characterize imprecise numerical information such as "about 7" or "approximately less than 7". A fuzzy number can be expressed in some membership function forms. Two important and widely used membership functions are linear triangular and linear trapezoidal. Figure 3.6 and 3.7 illustrate these membership functions, respectively.

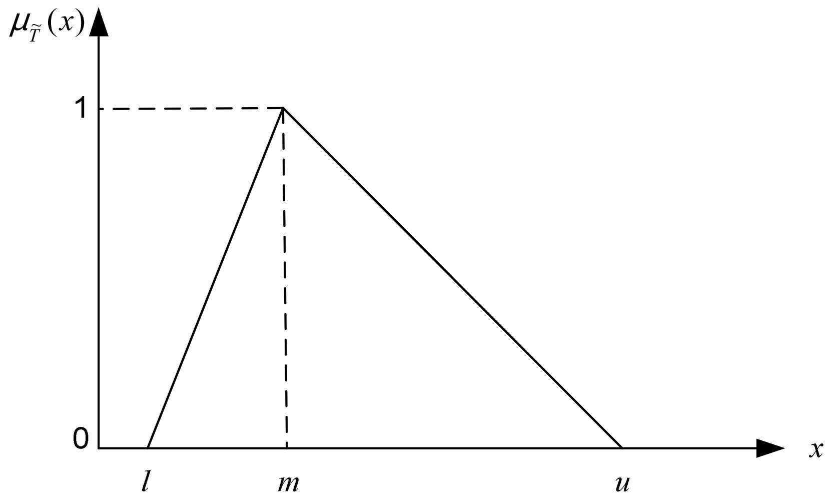


Figure 3.6 Triangular fuzzy number

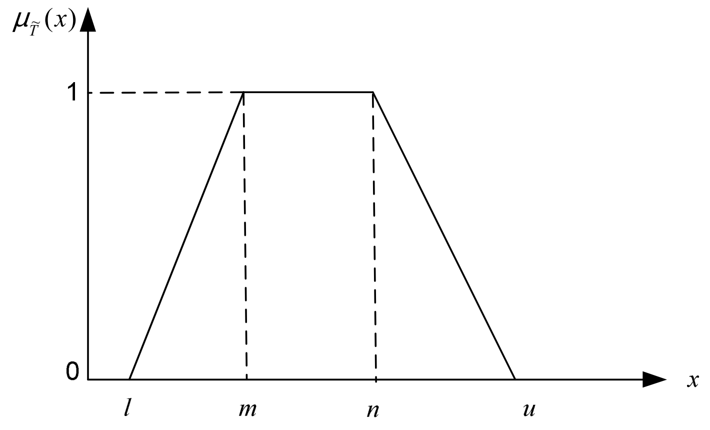


Figure 3.7 Trapezoidal fuzzy number

3.2.3 Fuzzy Mathematical Programming

When modeling a problem with multiple objectives, estimating exact values of the coefficients, the right hand side values of constraints, the target values of goals are difficult tasks. Even if all information can be provided by a decision maker, the uncertainty may still exist in the problem. Many researchers considered these type of problems with fuzzy mathematical programming methods.

Inuiguchi and Ramik (2000) classified the fuzzy mathematical programming methods into three categories considering the kinds of uncertainties treated in the method:

- Fuzzy mathematical programming with vagueness: it treats decision making problem under fuzzy goals and constraints,
- Fuzzy mathematical programming with ambiguity: it treats ambiguous coefficients of objective functions and constraints but does not treat fuzzy goal and constraints,
- Fuzzy mathematical programming with vagueness and ambiguity: it treats ambiguous coefficients as well as vague decision maker's preference.

There are a lot of fuzzy mathematical programming types. In this research, IFGP is employed to handle the Heterogeneous Vehicle Routing Problem with and without split deliveries. Therefore, only the three types of fuzzy mathematical programming methods, FLP, FGP and IFGP, are explained under this section.

3.2.2.1 Fuzzy linear programming

Consider an LP model,

$$\begin{array}{ll} \textit{minimize} & z = cx \\ \textit{subject to} & Ax \leq b \\ & x \geq 0 \end{array} \tag{3.22}$$

where c= (c₁, c₂, ..., cₙ) is the ***n*** dimensional row vector of coefficients of objective function, ***x*** is an ***n***-dimensional column vector of the decision variables, ***A*** is an ***m*** *x* ***n*** matrix of constants, and ***b*** is an ***m***-dimensional column vector of right-hand side constants. According to Zimmermann (1978), fuzzy version of the model (3.22) can be adopted as follows;

$$\left.\begin{array}{l} cx \prec z_0 \\ Ax \prec b \\ x \geq 0 \end{array}\right\} \tag{3.23}$$

where the symbols "$\prec$ and $\succ$" denote the fuzzified versions of "$\leq$ and $\geq$" and can be read as "essentially less (greater) than or equal to".

Zimmermann (1978) defined a linear membership function, $\mu_1(cx)$ for the goal as follows:

$$\mu_1(cx) = \begin{cases} 1 & if \quad cx \le z_0, \\ 1-(cx-z_0)/d_1 & if \quad z_0 \le cx \le z_0 + d_1, \\ 0 & if \quad cx \ge z_0 + d_1 \end{cases} \tag{3.24}$$

Zimmerman (1978) also proposed a linear membership function $\mu_{2i}(a_i X)$ to treat the i^{th} fuzzy constraint as follows:

$$\mu_{2i}((Ax)_i) = \begin{cases} 1 & if \quad a_i x \le b_i \\ 1-(a_i x - b_i)/d_{2i} & if \quad b_i \le a_i x \le b_i + d_{2i} \\ 0 & if \quad a_i x \ge b_i + d_{2i}, \end{cases} \tag{3.25}$$

Where *d_1* and *d_{2i}* (*i=1,2,...,m*) are chosen constants of admissible violations of the goal and the set of constraints, respectively (Mohamed, 1997). $\mu_1(cx)$ and $\mu_{2i}((Ax)_i)$ denote the degree of the membership of goals and constraints. It is assumed that the i^{th} membership function should be 1 if the i^{th} constraint is very well satisfied, 0 if the i^{th} constraint strongly violated its limit *d_{2i}*, and linear from 0 to1 (Sakawa, 1993). Figure 3.8 illustrates the "essentially less than or equal to" type linear membership function.

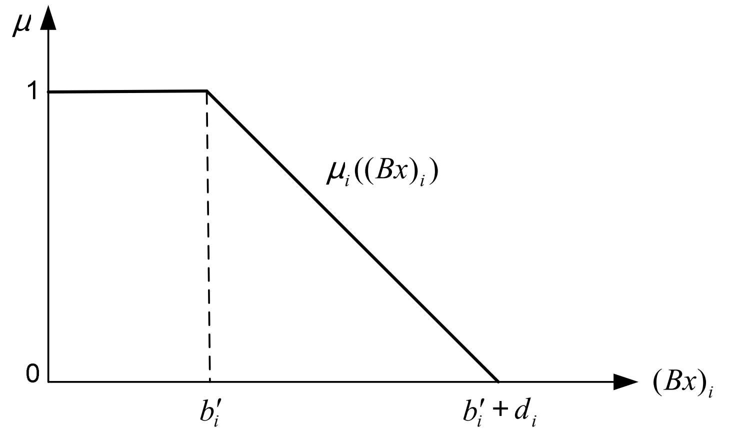


Figure 3.8 "$\prec$" type linear membership function

The degree of the membership of goals and constraints express the satisfaction of the decision maker with the solution, so membership functions value must be maximized (Mohamed, 1997).

After defining the linear membership functions, the maximizing decision is then defined by using the fuzzy decision theorem of Bellman and Zadeh (1970):

$$\max_x \min(\mu_1(cx),...,\mu_{21}(a_1x),...,\mu_{2m}(a_mx)) \qquad (3.26)$$

Introducing one new variable λ, this problem can be transformed as:

$$\begin{array}{ll} \max & \lambda \\ \textit{subject to} & \mu_1(cx) \geq \lambda \\ & \mu_{2i}(a_ix) \geq \lambda \quad i = 1,2,...,m \\ & x \geq 0 \end{array} \qquad (3.27)$$

According to above membership functions, FLP for (3.22) can be rewritten as following (Mohamed, 1997):

$$\begin{array}{ll} \max & \lambda \\ \textit{subject to} & \lambda \leq 1-(cx-z_0)/d_1 \\ & \lambda \leq 1-(a_ix-b_i)/d_{2i} \quad i = 1,2,...m \\ & \lambda \geq 0, x \geq 0. \end{array} \qquad (3.28)$$

FLP model can be easily extended to fuzzy multi-objective linear programming by defining a membership function for each of objective functions. Assume that there are *k* linear objective functions to be minimized; the corresponding model can be defined as

$$\begin{array}{lll} \max & \lambda & \\ \textit{subject to} & \lambda \leq 1-(c_kx-z_{0k})/d_{1k} & k = 1,2,...,k \\ & \lambda \leq 1-(a_ix-b_i)/d_{2i} & i = 1,2,...,m \\ & \lambda \geq 0, x \geq 0. & \end{array} \qquad (3.29)$$

3.2.2.2 Fuzzy Goal Programming

Goal programming (GP) is one of the most powerful multi-objective decision making approaches. A standard GP formulation requires that the target values of the goals and the parameters of the constraints are precisely known. However, one of the major drawbacks for a decision maker in GP is to determine the goal value of each objective function precisely.

The main idea behind GP is to minimize the distance between the objective value Z_k and an aspiration level (target value of the objective function) $\overline{Z}_k$, which is expressed by the deviational variables. In FGP, membership function values of each objective are replaced by the deviational variables (Mohamed, 1997).

In general, a typical FGP problem can be formulated as follows:

Find $x_i \qquad i = 1,2,...,n$

to satisfy

$$\begin{aligned} &Z_m(x_i) \prec \overline{Z}_m && m = 1,2,...,M, \\ &Z_k(x_i) \succ \overline{Z}_k && k = M+1, M+2,...,K, \\ &g_j(x_i) \leq b_j && j = 1,2,...,J, \\ &x_i \geq 0 && i = 1,2,...,n. \end{aligned} \tag{3.30}$$

where

$Z_m(x_i)$ = the mth goal constraint,

$Z_k(x_i)$ = the kth goal constraint,

$\overline{Z}_m(x_i)$ = the target value of the mth goal,

$\overline{Z}_k(x_i)$ = the target value of the kth goal,

$g_j(x_i)$ = the jth inequality constraint,

b_j = the available resource of inequality constraint j.

In formulation (3.30), the symbols "$\prec$" and "$\succ$" denote the fuzzified versions of "$\leq$" and "$\geq$" and can be read as "approximately less (greater) than or equal to". These two types of linguistic terms have different meanings. Under "approximately less than or equal to" situation, the goal *m* is allowed to be spread to the right-hand-side of $\overline{Z}_m$ ($\overline{Z}_m = l_m$ where l_m denote the lower bound for the m^{th} objective) with a certain range of r_m ($\overline{Z}_m + r_m = u_m$, where u_m denote the upper bound for the m^{th} objective). Similarly, with "approximately greater than or equal to", Z_k is allowed to go to the left side of $\overline{Z}_k$ ($\overline{Z}_k$ - $p_k = l_k$, and $\overline{Z}_k = u_k$) (Wang and Fu, 1997).

After constructing fuzzified aspiration levels with respect to the linguistic terms of "approximately less than or equal to", and "approximately greater than or equal to", appropriate fuzzy membership function can be developed for each goal as follows:

For "approximately less than or equal to";

$$\mu_{z_m}(x) = \begin{cases} 1 & if \quad Z_m(x) \leq l_m, \\ 1 - \dfrac{Z_m(x) - l_m}{u_m - l_m} & if \quad l_m \leq Z_m(x) \leq u_m, \\ 0 & if \quad Z_m(x) \geq u_m. \end{cases} \tag{3.31}$$

For "approximately greater than or equal to";

$$\mu_{z_k}(x) = \begin{cases} 1 & if \quad Z_k(x) \geq u_k, \\ 1 - \dfrac{u_k - Z_k(x)}{u_k - l_k} & if \quad l_k \leq Z_k(x) \leq u_k, \\ 0 & if \quad Z_k(x) \leq l_k. \end{cases} \tag{3.32}$$

Figure 3.8 illustrates both types of membership functions.

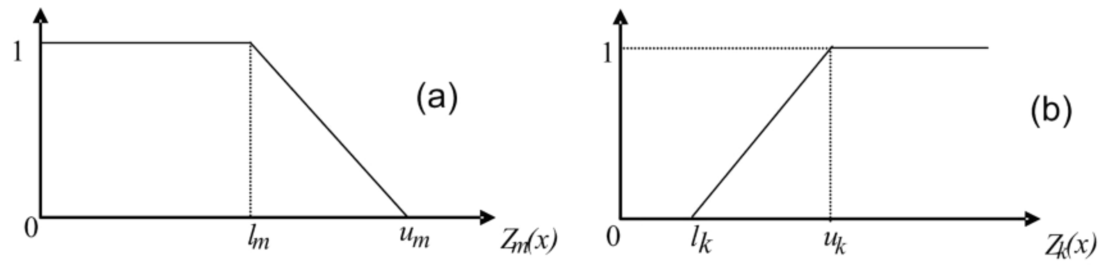


Figure 3.8 Membership functions of fuzzy goals, (a) “approximately less than or equal to” (b)“approximately greater than or equal to”.

Using Bellman and Zadeh (1970)’s fuzzy decision theorem, the fuzzy solution is obtained by the intersection of the all the membership functions representing the fuzzy goals. The membership function $\mu_F(x)$ which characterizes the fuzzy solution can be defined as follows (Sakawa, 1993):

$$\mu_F(x) = \mu_{Z_1}(x) \cap \mu_{Z_2}(x) .... \cap \mu_{Z_k}(x) = \min[\mu_{Z_1}(x), \mu_{Z_2}(x), ..., \mu_{Z_k}(x)] \qquad (3.33)$$

Then the optimum decision is one that maximizes the minimum membership function values (Sakawa, 1993):

$$\max_{x \in F} \mu_F(x) = \max_{x \in F} \min[\mu_{Z_1}(x), \mu_{Z_2}(x), ..., \mu_{Z_k}(x)] \qquad (3.34)$$

By introducing the auxiliary variable λ, formulation (3.30) can be transformed to:

$$\left.\begin{array}{lll} \text{maximize} & \lambda & \\ \text{subject to} & & \\ & \lambda \le \mu_{Z_k} & k = 1,...,K \\ & g_j(x_i) \le b_j & i = 1,..., n, \quad j = 1,..., J \\ & x_i \ge 0 & i = 1,..., n \\ & \lambda \in [0,1]. & \end{array}\right\} \qquad (3.35)$$

3.2.2.3 Interactive Fuzzy Goal Programming

In the FLP and FGP approaches discussed in the above sections, the fuzzy decision of Belman and Zadeh (1970) is used to present the fuzzy preferences of the decision maker. Sakawa (1993) stated that the use of the fuzzy decision may not be appropriate in practice and consequently it becomes evident that an interaction with the decision maker is necessary. Also it is pointed out by Sakawa (1993) that fuzzy mathematical programming approaches can be strengthened by incorporating the desirable features of the interactive approaches into fuzzy approaches.

If the decision maker is not satisfied with the current optimal solution, IFGP approaches allows the decision maker to act on this solution by updating the membership functions (Abd El-Wahed and Lee, 2006).

Abd El-Wahed and Lee (2006) stated that the main advantage of interactive approaches is that the decision maker controls the search directions during the solution procedure until a preferred compromise solution is obtained.

The solution procedure of IFGP which belongs to El-Wahed and Lee (2006) is given in the following:

"Step 1: Develop a multi-objective linear programming model.

Step 2: Solve the first objective function as a single objective problem. Continue this process K times for the K objective functions. If all the solutions are the same, select one of them as an optimal compromise solution and go to Step 8. Otherwise, go to Step 3.

Step 3: Evaluate the objective function at the Kth solution and determine the best lower bound (l_k) and the worst upper bound (u_k).

Step 4: Define the membership function of each objective function and also the initial aspiration level.

Step 5: Use model formula that is given in (3.35) and solve it as a linear programming problem.

Step 6: Present the solution to the decision maker. If the decision maker accepts it, go to Step 8. Otherwise, go to Step 7.

Step 7: Evaluate each objective function of the solution. Compare the upper bound of each objective with the new value of the objective function. If the new value is lower than the upper bound, consider this as a new upper bound. Otherwise, keep the old one as is. Repeat this process K times and go to Step 4.

Step 8: Stop."

The IFGP procedure used in this research is slightly different than from the one described by El-Wahed and Lee (2006). In their approach, the iterations stop when the decision maker is satisfied with the solution. In the proposed approach, iterations go on automatically until the difference in the objective function value of the last two iterations is less than an expected value. The IFGP procedure used in the proposed methodology is discussed in detail in Chapter 4 of this dissertation.

CHAPTER FOUR
THRESHOLD ALGORITHM

In real life distribution problems, most of the companies own a heterogeneous fleet of vehicles. If this is the case, the distribution planning problem is called heterogeneous vehicle routing problem (HVRP). The purpose of this chapter is to propose an algorithm specifically designed for HVRPs considering the special characteristics of the problem.

The algorithm proposed in this study is a cluster first route second type of algorithm (Laporte and Semet, 2002). These types of algorithms are based on the idea of splitting a large size problem into subproblems and solving them faster. The success of these algorithms mainly depends on how well the clusters are formed. Throughout this research, in order to form effective clusters, a distance threshold level is used together with an interactive fuzzy goal programming (IFGP) approach. Then each cluster is solved to optimum by constraint programming (CP).

The clustering phase is based on a distance threshold level and it is increased gradually at every iteration until a solution is achieved which is satisfactory for the decision maker. Therefore, the proposed approach will be referred as the "*Threshold Algorithm*" throughout the study.

In the first section of the chapter, Threshold Algorithm is explained in detail. Then the performance of the proposed algorithm is tested on a group of benchmark problems in the literature. After performance tests, Threshold Algorithm is employed to solve the fresh goods distribution problem of a national retail chain store.

4.1 Proposed Algorithm

In the clustering phase of the proposed approach, integer programming (IP) is used to solve a set covering problem. Then, from the sets selected, clusters are formed and vehicles are assigned through an IFGP approach. In the finalization of the proposed approach, subproblems are solved through constraint programming (CP). The flow of the proposed algorithm can be seen in Figure 4.1 (Mizrak Ozfirat, Ozkarahan, 2007).

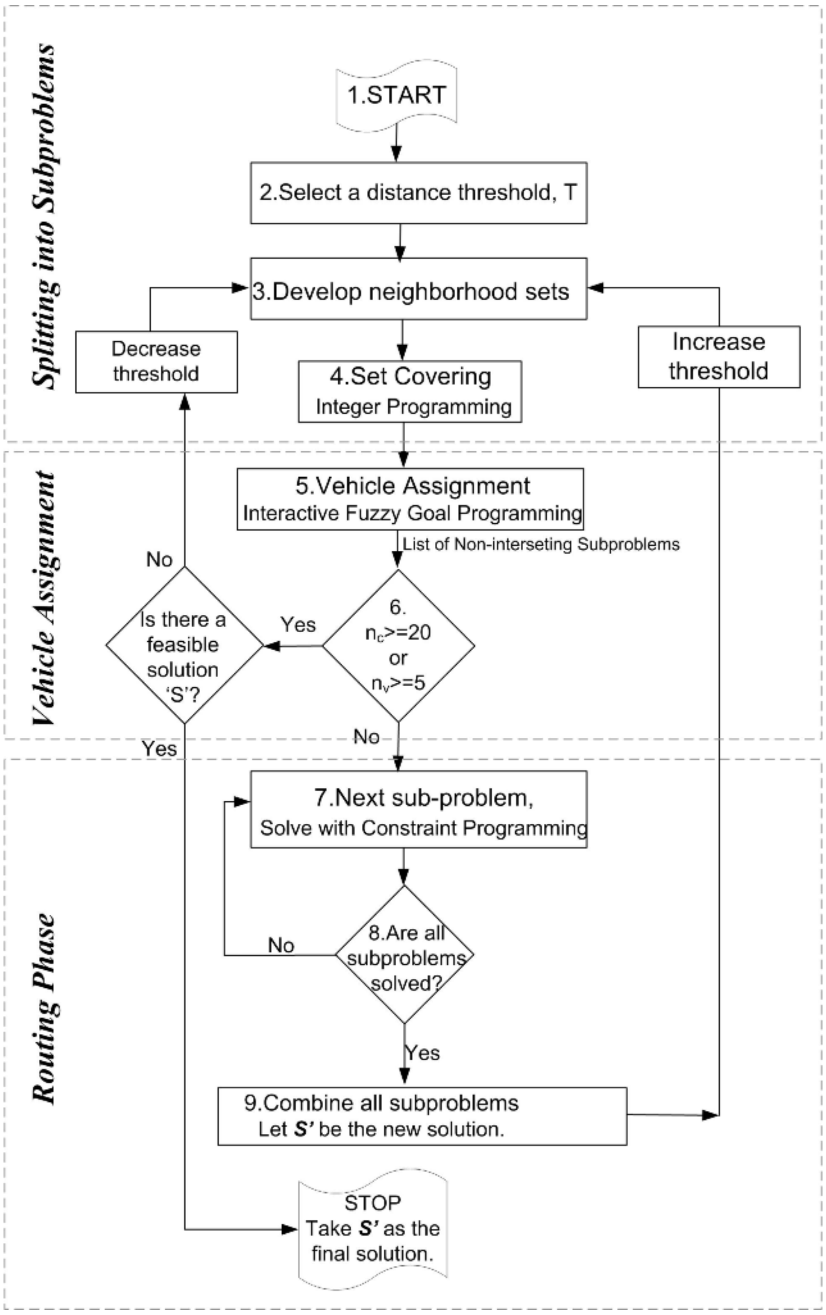

Splitting into Subproblems
1.START
2.Select a distance threshold, T
3.Develop neighborhood sets
Decrease threshold
4.Set Covering Integer Programming
Increase threshold
Vehicle Assignment
5.Vehicle Assignment Interactive Fuzzy Goal Programming
List of Non-interseting Subproblems
No
Is there a feasible solution 'S'?
Yes
6. n_c>=20 or n_v>=5
Yes
No
Routing Phase
7.Next sub-problem, Solve with Constraint Programming
No
8.Are all subproblems solved?
Yes
9.Combine all subproblems Let S' be the new solution.
STOP Take S' as the final solution.

Figure 4.1 Flow diagram of the proposed algorithm(n_v: Number of vehicles; n_c: Number of nodes)

4.1.1 Splitting into Subproblems

The first part of the algorithm is splitting the main problems into clusters. The algorithm is given in the following.

Step 1: Start

Step 2: Select a threshold ***T***. Let there be ***"n_c"*** customers to be served.

Step 3: Develop the neighborhood sets of each customer within distance ***T*** to the customer. In other words, a set belonging to each customer covering all other customers, which are at most ***"T"*** km. away from itself, is developed. This makes up totally ***"n_c"*** sets (A sample is given in Figure 4.2).

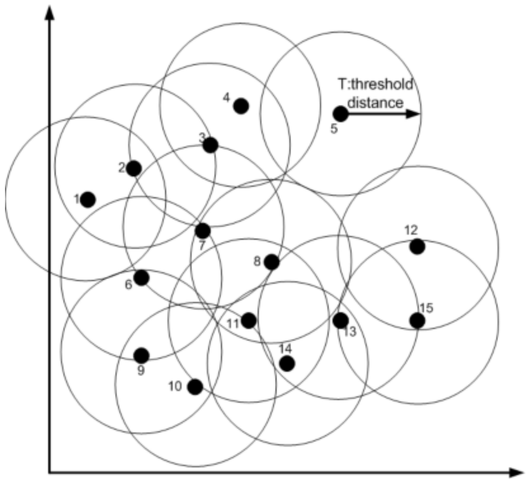


Figure 4.2 Let n_c=15, the 15 sets developed within distance *T* to each customer

Step 4: Set Covering: In this phase, among the ***"n_c"*** sets developed in Step 3, a number of them are selected to cover all customers in the most centralized way. In other words the neighborhood points in the sets selected are closer to the center of their neighborhood. An IP model, called Model SC given in Formulation 4.1, is used in this step. The notation used in the model is given in Table 4.1. (OPL Code is given in Appendix A).

Table 4.1. Notation used in Model SC

Parameters :
n_c : *Number of nodes to be served.*
D_{ij} : *Distance from node i to node* $j, i : 0..n_c, j : 0..n_c$.
$M_{ij} : \begin{cases} 1 \text{ if set } i \text{ covers node } j. \\ 0 \text{ otherwise} \end{cases} \quad , i : 1..n_c, j : 1..n_c.$
Sets :
$Nodes = \{1,2,.....,n_c\}$
Decision Variables :
$t_i : \begin{cases} 1 \text{ if set } i \text{ is selected to be a subproblem} \\ 0 \text{ otherwise} \end{cases} \quad , i{:}1..n_c$

$$
\begin{aligned}
&Model\ SC: \\
&Min\ z_1 = \sum_{j \in Sets} \sum_{i \in Sets} M_{ij} D_{ij} t_i && (4.1a) \\
&subject\ to && \quad \mathbf{(4.1)} \\
&\sum_{i \in Sets} M_{ij} t_i >= 1 \qquad \forall j \text{ in } Nodes && (4.1b)
\end{aligned}
$$

According to the objective function ***(4.1a)*** the sets selected are not arbitrarily the minimum number of sets but rather, the minimum number of sets which are most centralized. Constraints ***(4.1b)*** state that all nodes should be covered at least once.

With the objective function ***(4.2)***, the total number of sets selected can be minimized. However, instead of ***(4.2)***, objective function ***(4.1a)*** is used where the distances between the nodes of selected sets are considered.

$$z = \sum_{i \in Sets} t_i \qquad \mathbf{(4.2)}$$

To state the difference in between the two objective functions, an example problem is given in the following:

Sample Problem: Let there be 4 nodes. Let distance matrix D and coverage matrix M be as follows.

Matrix M where,

$$M_{ij} : \begin{cases} 1 \text{ if set } i \text{ covers customer } j. \\ 0 \text{ otherwise} \end{cases}$$

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Distance Matrix D where,

D_{ij}: Distance from node i to node j.

$$D = \begin{bmatrix} 0 & 40 & 25 & 5 \\ 40 & 0 & 15 & 45 \\ 25 & 15 & 0 & 32 \\ 5 & 45 & 32 & 0 \end{bmatrix}$$

Looking at matrix M, the neighborhood sets of the four nodes are as follows:

$$Set1 = \{1,3,4\}; \quad Set2 = \{2,3\}; \quad Set3 = \{1,2,3\}; \quad Set4 = \{1,4\}$$

In Model SC, if D_{ij} is not considered in the objective function as in ***(4.2)***, one of the following solutions would cover all nodes:

- Set 1 and Set2
- Set 1 and Set 3
- Set 2 and Set 4
- Set 3 and Set 4

However when D_{ij} is included, then:

If $t_1 = 1$ then $\Sigma M_{1j} D_{1j} t_1 = 30$

If $t_2 = 1$ then $\Sigma M_{2j} D_{2j} t_2 = 15$

If $t_3 = 1$ then $\Sigma M_{3j} D_{3j} t_3 = 40$

If $t_4 = 1$ then $\Sigma M_{4j} D_{4j} t_4 = 5$

Hence the solution would be:

- Set 2 and Set 4 where $\Sigma M_{ij} D_{ij} t_i = 15+5=20$

A sample output of Model SC (when the input is the problem in Figure 4.2) is given in Figure 4.3.

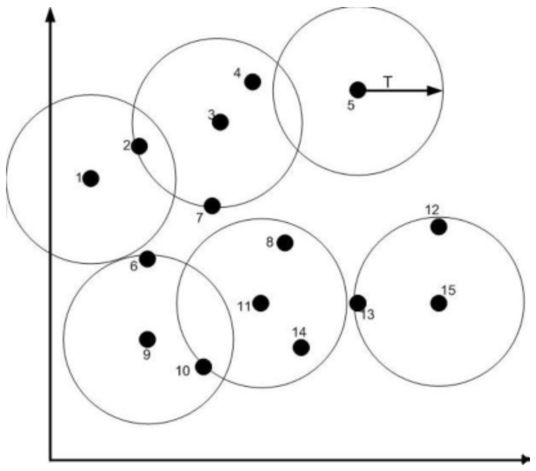


Figure 4.3 The selected sets to make the number minimum and to cover all customers

4.1.2 Vehicle Assignment

Step 5: Vehicle Assignment: The sets selected in Step 4 may be intersecting. These should be turned into non-intersecting sets, and the necessary vehicles should be assigned to each set in order to develop subproblems. When assigning the vehicles their fixed costs should be taken into consideration. However, considering fixed costs without thinking of its effect on the traveling cost would bring out inefficient solutions.

That is, as the capacity of a vehicle increases, the fixed cost also increases. If we try to minimize total fixed cost in the assignment of vehicles to subproblems, smaller vehicles in capacity would be selected. Hence, the number of vehicles would increase (total number of tours visiting the depot increases) and total distance traveled would increase accordingly.

On the other hand, if the objective function is minimizing total number of vehicles (instead of fixed cost) in the vehicle assignment, then the algorithm would tend to select higher capacity but more expensive vehicles. In this case total traveling cost would decrease but total fixed cost would increase. The tradeoff between fixed cost and traveling cost can be seen in Figure 4.4.

Objective: Number of vehicles is minimized	Total Fixed Cost ↑	Total Traveling Cost ↓
Objective: Total fixed cost of vehicles is minimized	Total Fixed Cost ↓	Total Traveling Cost ↑

Figure 4.4 Tradeoff between total fixed cost and total traveling cost

Therefore in vehicle assignment, the conflict between these two cost terms should be worked out. When there exist multiple objectives in a problem and the degree of fulfillment of these objectives is vague, fuzzy mathematical programming may be a useful tool to handle the problem (Zimmerman, 1978). Therefore in the proposed approach, a novel IFGP approach is built.

The IFGP approach considers the two objective functions, which are minimizing total fixed cost and minimizing total number of vehicles, together. A fuzzy goal programming (FGP) model is built which adds these two objective functions as fuzzy constraints. Then, this model is solved iteratively until the difference between two consecutive iterations satisfy a prespecified rule. The main difference of this approach from other IFGP approaches in the literature is that it is independent from the decision maker. In other words, IFGP approach works not to satisfy the decision maker but to satisfy a prespecified state of the problem. In addition, to the best of our knowledge, this approach is the first one which adds fuzziness into HVRP. The proposed IFGP approach is explained detailly in the following.

The formulation of the FGP model, which is called the Model VA, is given in Formulation 4.5. The notation used in Model VA is given in Table 4.2 (OPL code of the model is given in Appendix A).

Table 4.2 Notation used in the vehicle assignment model

Parameters :
n_c : *Number of nodes to be served.*
n_v : *Number of vehicles.*
$Demand_j$:*Demand of node* $j, j : 0..n_c$.
Cap_v : *Capacity of vehicle* $v, v : 1..n_v$.
$Cost_v$: *Fixed cost of vehicle* $v, v : 1..n_v$.
$z_{1\min}$:*Minimum number of vehicles.*
$z_{2\min}$:*Minimum fixed cost*
d_1:*Difference between the aspiration level and maximum level of number of vehicles*
d_2:*Difference between the aspiration level and maximum level of fixed cost*
Sets :
$Nodes = \{1,2,.....,n_c\}$
$T^* = \{t_i^* : i \in Nodes\}$: *Optimum solution of the set covering model.*
$Subproblems = \{i \in Nodes : t_i^* = 1\}$.
$Vehicles = \{1,2,.....,n_v\}$
Decision Variables :
$y_{ij} : \begin{cases} 1 \text{ if customer } j \text{ is assigned to subproblem } i \\ 0 \text{ if customer } j \text{ is not included in subproblem } i \end{cases} \quad ,i \in Subproblems \; ,j \in Nodes.$
$w_{iv} : \begin{cases} 1 \text{ if vehicle } v \text{ is assigned to subproblem } i \\ 0 \text{ otherwise} \end{cases} \quad ,i \in Subproblems \; ,v{:}1..n_v.$
λ : *Auxillary variable of the fuzzy goal programming model.*

The two goals considered are:

- To minimize total number of vehicles,

$$z_1 : Min.\sum_i \sum_v w_{iv} \tag{4.3}$$

- To minimize total fixed cost of vehicles,

$$z_2 : Min. \sum_i \sum_v Cost_v w_{iv} \tag{4.4}$$

$$\begin{aligned}
&\mathit{Model\ VA}: \\
&\mathit{Min.}\ \ \lambda && (4.5a) \\
&\mathit{subject\ to} \\
&\lambda \le 1 - \frac{\sum_i \sum_v w_{iv} - z_{1\min}}{d_1} && (4.5b) \\
&\lambda \le 1 - \frac{\sum_i \sum_v \cos t_v * w_{iv} - z_{2\min}}{d_2} && (4.5c) \\
&0 \le \lambda \le 1 && (4.5d) \\
&\sum_{i \in Subproblems} y_{ij} = 1 \qquad \forall j \in Nodes && (4.5e) \\
&y_{ij} \le M_{ij} \qquad \forall i \in Subproblems, j \in Nodes && (4.5f) \\
&\sum_{v \in Vehicles} Cap_v w_{iv} \ge \sum_{j \in Nodes} y_{ij} Demand_j \qquad \forall i \in Subproblems && (4.5g) \\
&\sum_{i \in Subproblems} w_{iv} \le 1 \qquad \forall v \in Vehicles && (4.5h) \\
&w_{iv} \in \{0,1\} \qquad \forall i \in Subproblems, \forall v \in Vehicles && (4.5i) \\
&y_{ij} \in \{0,1\} \qquad \forall i \in Subproblems, \forall j \in Nodes && (4.5j)
\end{aligned} \qquad \mathbf{(4.5)}$$

where:

z_{1min} : Aspiration level of goal 1.

z_{2min}: Aspiration level of goal 2.

d_1: Limit acceptable to exceed aspiration level of goal 1.

d_2: Limit acceptable to exceed aspiration level of goal 2.

The objective function ***(4.5a)*** belongs to fuzzy approach and the constraints ***(4.5b)***, ***(4.5c)*** and ***(4.5d)*** are fuzzy constraints. In addition to fuzzy constraints, system constraints appear in the model with ***(4.5e)*** to ***(4.5j)***. The model assigns each customer to only one of the sets (found in Step 4) in which it appears. This is stated in the model by constraints ***(4.5e)*** and ***(4.5f)***. Constraints ***(4.5g)*** assure that the total capacity of vehicles assigned to a subproblem must be greater than or equal to the total demand of customers in that subproblem. Constraint ***(4.5h)*** states that a vehicle can only be allocated to one subproblem. Finally, binary decision variable constraints are given by ***(4.5i)*** and ***(4.5j)***. The output of the model determines the subproblems as well as the vehicles assigned to each one to have a number of NP-complete subproblems.

4.1.2.1 Setting Fuzzy Parameters of the Interactive Fuzzy Goal Programming Approach: z_{1min}, z_{2min}, d_1, d_2

Firstly, Model VA is solved only with the system constraints (constraints 4.5e to 4.5j) for the objective functions z_1 and z_2 (Equation 4.3 and 4.4) respectively. Let the minimum values of these objectives be z_{1min} and z_{2min} respectively. That is, z_{1min} is the minimum number of vehicles necessary and z_{2min} is the minimum vehicle fixed cost. The aspiration levels of these two goal are set to be z_{1min} and z_{2min} respectively. Let the corresponding fixed cost to z_{1min} (minimum number of vehicles) be z_2^0 and the corresponding number of vehicles to z_{2min} (minimum fixed cost) be z_1^0.

Then, fuzzy constraints 4.5b, 4.5c and 4.5d are added to Model VA. The limit which is acceptable to exceed total number of vehicles (d_1) is set to be 1 and the limit on total fixed cost (d_2) is set to be ($z_2^0 - z_{2min}$).

Then the fuzzy model is solved to give the corresponding fixed cost of $z_{1min}+1$ vehicles. Let the fixed cost of this solution be z_2^1. In the next iteration, the limit on the number of vehicles is increased by one. The new limit on the fixed cost is set to be z_2^1- z_{2min}. The model is resolved with $z_{1min}+2$ vehicles to give the fixed cost of this solution which is z_2^2. The iterations go on in this manner.

The difference between the fixed costs of two consecutive iterations is a negative number. In other words, at every iteration, the fixed cost will decrease by $(z_2^{i+1} - z_2^i)$.

However, as fixed cost decreases, number of vehicles and hence the traveling costs are expected to increase (The relation between the fixed cost and the number of vehicles can be seen in Figure 4.5). The expected increase in the traveling cost when the number of vehicles is increased by one is called the ***ExpectedLoss***.

Throughout the iterations of IFGP, at a certain point, the expected increase in the traveling cost (***ExpectedLoss***) would be higher than the decrease in the fixed cost. After this point the total cost would start increasing. Therefore, the iterations stop there and the solution of the ith iteration is accepted. Since the FGP model is operated iteratively until a certain condition is satisfied, it is an interactive FGP algorithm. The flow of the algorithm can be seen in the following.

IFGP Algorithm:

```
1. Start;
2. Solve Model VA with constraints 4.5e to 4.5j to minimize $z_1$ and $z_2$ respectively;
3. $z_{1min}$ :=minimum of $z_1$;
   $z_{2min}$ :=minimum of $z_2$;
4. $z_2^0$:=corresponding cost of $z_{1min}$;
   $z_1^0$:=corresponding number of vehicles of $z_{2min}$;
5. $d_1$:= 1 ;      $d_2$:= $z_2^0 - z_{2min}$;
6. Solve Model VA;
   $z_2^i$:=Total Fixed Cost;
7. If $(z_2^i - z_2^{i-1})$>= ExpectedLoss, then 8; else 10.
8. $d_1$:= $d_1+1$; $d_2$:= $z_2^i - z_{2min}$;
9. Go to 6;
10. Stop;
```

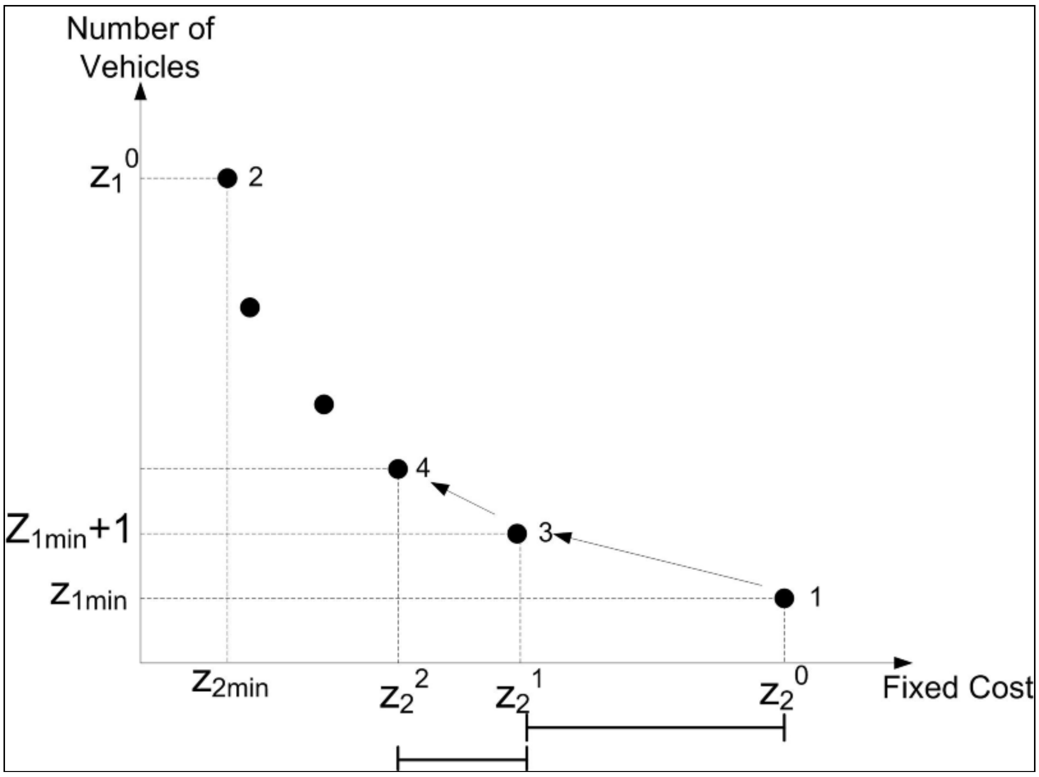


Figure 4.5 The model is solved iteratively until $(z_2^{i+1} - z_2^i) < ExpectedLoss$.

To explain the operation of the IFGP approach more clearly, Figure 4.5 is given. At the very beginning, points 1 and 2 on the graph are found. (Point 1 is found by minimizing z_2 and Point 2 is found by minimizing z_1.) Then the iterations start from point 1 and follow the order of points 3, 4, and so on. At iteration ***i+1***, when

$$(z_2^{i+1} - z_2^{i}) < ExpectedLoss \qquad (4.6)$$

the procedure stops. The solution achieved in iteration ***i*** is accepted. In the solution, the nodes of all subproblems as well as the vehicles assigned to them are obtained.

One important question in this procedure is how to set ***ExpectedLoss***. After a number of experiments, this value is assumed to be the maximum of distances of all nodes to the depot node.

In order give a better view of IFGP, a sample problem is provided in the following.

Sample Problem: Let ***ExpectedLoss*** be 37. Let the results of minimizing the two goals be as in Table 4.3.

Table 4.3 Sample problem data

	Number of vehicles	**Fixed Cost of Vehicles**
Minimize Z_1	$z_{1min} = 6$	1945
Minimize Z_2	42	$z_{2min} = 1050$

When the fuzzy model is solved iteratively, the solutions in the Table 4.4 are achieved:

Table 4.4 Iterations of the fuzzy model, sample problem

Iteration No	**Number of Vehicles**	**Fixed Cost of Vehicles**	**Difference Between The Last Two Iterations**
0	6	1945	-
1	7	1895	50
2	**8**	**1855**	**40**
3	9	1820	35

It can be seen from Table 4.4 that the decrease in fixed cost at iteration 1 is 50 units. At second iteration the decrease turns out to be 40 and at third iteration, it is 35. Since the ***ExpectedLoss*** value (expected increase in traveling cost as number of vehicles decrease by one) is assumed to be 37, iterations stop at third one (Decrease in fixed cost=35<37=***ExpectedLoss***). Then the results of iteration number 2 is selected.

Step 6: When the main problem is decomposed into subproblems, each one will be handled on its own. However, there is a drawback of this algorithm. Every time the threshold is increased, the subproblems get larger in size. Hence, the solution time increases considerably. The expected behavior of solution time against the threshold level can be seen in Figure 4.6.

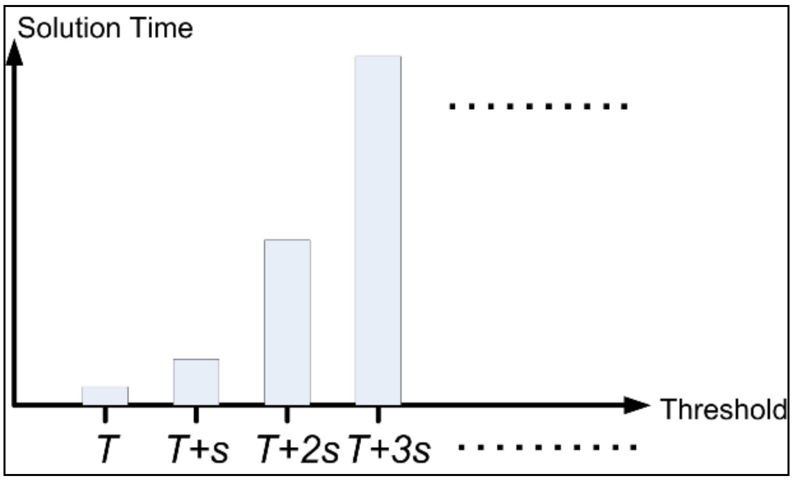


Figure 4.6 Expected behavior of solution time against threshold level. As threshold increases, solution time is expected to increase quadratic ally

After a certain level of threshold ***T***, the optimal solution cannot be found with the existing technology. Therefore stop the algorithm at that level of ***T***. Take the existing solution.

The level to stop the algorithm is found by trial and error method. Through a number of experiments, that level is determined to be n_c >= 20 (number of nodes) or n_v >= 5 (number of vehicles). In other words if n_c >= 20 or n_v >= 5 for any one of the subproblems then the procedure STOPS. The solution achieved in the previous iteration is accepted. However, if no solution is achieved at the very first iteration, then the procedure goes on by lowering the threshold and to Step 2.

4.1.3 Routing Phase

Step 7: This step is the routing phase of the algorithm. In order to solve the subproblems obtained in Step 5, a CP model, called the Model CPM, is built and each subproblem is solved by this model. The Model CPM is given in Formulation 4.7. (OPL code of the model is given in Appendix A). Notation used in Model CPM is given in Table 4.5.

Table 4.5 Notation used in model CPM

Parameters:
n_c : *Number of nodes to be served.*
n_v : *Number of vehicles.*
$Demand_j$: *Demand of node j.*
Cap_v : *Capacity of vehicle v.*
Sets:
$Nodes = \{1,2,.....,n_c\}$
$Y^* = \{y^*_{ij} : i \in Subproblems, j \in Nodes\}$ *:Optimum solution of Model VA.*
$Nodes_k = \{j \in Nodes : y^*_{kj} = 1\} \quad \forall k \in Subproblems; Let \ \lvert Nodes_k \rvert = a_k \quad \forall k \in Subproblems$
$Total_k = Nodes_k \cup \{0\}$; "0" *denotes depot node.*
$Vehicles = \{1,2,.....,n_v\}$
Decision Variables:
$X_{ij} : \begin{cases} \text{Number of the vehicle that travels from node } i \text{ to node } j. \\ 0 \quad \text{if no vehicle travels from node } i \text{ to node } j. \end{cases}$

Model CPM :

Solve

$$\sum_{j \in Total_k} (X_{ij} > 0) = 1 \qquad \forall i \in Nodes_k \qquad (4.7a)$$

$$\sum_{i \in Total_k} (X_{ij} > 0) = 1 \qquad \forall j \in Nodes_k \qquad (4.7b)$$

$$\sum_{i \in Total_k} (X_{i0} = v) = 1 \qquad \forall v \in Vehicles \qquad (4.7c) \qquad \mathbf{(4.7)}$$

$$\sum_{j \in Total_k} (X_{0j} = v) = 1 \qquad \forall v \in Vehicles \qquad (4.7d)$$

$$\sum_{j \in Total_k} X_{ij} = \sum_{j \in Total_k} X_{ji} \qquad \forall i \in Total_k \qquad (4.7e)$$

$$\sum_{i \in Total_k} \sum_{j \in Total_k, j \neq i} (X_{ij} = v) * Demand_j \le Cap_v \qquad \forall v \in Vehicles \qquad (4.7f)$$

Constraints ***(4.7a)*** and ***(4.7b)*** assure that each customer is visited exactly once. In addition, each vehicle should visit the depot once since the vehicles assigned to the problem are known in advance (from Model VA, given in Formulation 4.5). This is included in the model with constraints ***(4.7c)*** and ***(4.7d)***. Constraint set ***(4.7e)*** provides that a vehicle visiting a customer should leave that customer. Finally, it is stated by constraint set ***(4.7f)*** that capacity of vehicles should not be exceeded.

If a single vehicle is assigned to the subproblem, then it becomes a traveling salesman problem. In this case constraint set ***(4.7f)*** becomes invalid. However, if the subproblem requires more than one vehicle, than it is a VRP and all constraints are applicable. The model is written and solved in OPL Studio 3.7 (ILOG, 2003).

In Formulation 4.7, one can easily notice that objective function and the subtour elimination constraints are missing. A subtour is the tour of a vehicle without visiting the depot node. Without subtour elimination constraints, solutions as shown in Figure 4.7 can be obtained which are not desirable.

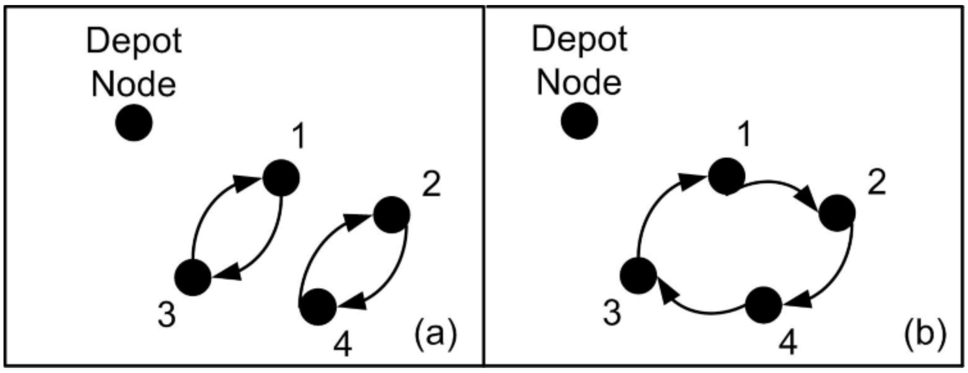


Figure 4.7 Examples of subtours

However, the number of these constraints increases non-polynomially as the number of nodes increase. Therefore, VRPs are NP-hard due to subtour elimination constraints. In the proposed approach, these are handled with an OPL Script algorithm (ILOG, 2003). The flow of the Subtour Elimination Algorithm (SEA) can be seen in the following.

OPL Script SEA:

```
1.    Start;
2.    MinimumDistance:=MaximumInteger;
3.    Let $S$ be the next solution of model CPM;
4.    If $S \neq \emptyset$ then 5, else 9;
5.    Distance:=Total Distance Traveled in $S$;
6.    If S is a full solution (does not contain sub-tours), then 7, else 8
7.    If Distance<MinimumDistance, then MinimumDistance:=Distance;
8.    Go to 3;
9.    Stop;
```

SEA written in OPL Script language handles the first possible solution of the constraint programming model (Model CPM). It controls whether the solution consists of a subtour. If it is a full solution (does not contain any subtours), the value of total distance traveled is recorded as ***MinimumDistance.*** Then the algorithm takes the next possible solution (of Model CPM) and makes subtour controls. If it is not a subtour and the total distance traveled belonging to this solution is less than ***MinimumDistance***, it is recorded as the new ***MinimumDistance***. The procedure goes on in the same way until all possible solutions are handled. Once enumeration is completed, the final ***MinimumDistance*** value turns out to be the optimum solution of the problem. The OPL code of the SEA is given in Appendix A.

- Speeding Up CP: OLP Studio offers several search procedures in order to decrease computation time of large scale models. These are Best First Search (BFS), Slice Based Search (SBS), Depth Bounded Discrepancy Search (DDS), Depth First Search (DFS), Interleaved Depth First Search (IDFS).

SBS and DDS are thought to be suitable to the nature of VRPs since there exist many good heuristics. Both of them are tested on several problems. SBS is found to be provide better computation times. Therefore, all computations are made using SBS.

Step 8: If all subproblems are finished then go to Step 9, else go to Step 7.
Step 9: Combine the solutions of all subproblems to give the solution of the main problem.
Step 10: Increase threshold by ‘***s***’ units, T:=T+***s***. Go to Step 3.

4.2 Tests On Sample Instances

To the best of our knowledge, there are no lower bounds for HVRP in the literature. Neither there exists an exact approach for all HVRP problems. The only way to test newly developed algorithms is to try them on benchmark instances given in the literature and compare the solutions with other known techniques.

In this study, ten instances are solved which are taken from Gendreau et al. (1999). They are also solved by Golden et al.(1984), Osman and Salhi (1996) (Gendreau et. al., 1999), Taillard (1999), Li et al. (2006) and Choi and Tcha (2007) . Golden et al.(1984) and Li et al. (2006) only considered total traveling cost, leaving total fixed cost of vehicles out. On the other hand, Osman and Salhi (1996) (Gendreau et. al., 1999), Taillard (1999), Gendreau et al. (1999), Choi and Tcha (2007) considered both traveling and fixed costs. Similar to these studies, both cost terms are considered in this research.

The benchmark instances employed are 20 node, 50 node and 75 node instances respectively. Distance values are all found analytically from the coordinates of customer nodes. The details of problem data are given in Table 4.6. Numbering schemes of the problems are similar to Gendreau et al.(1999). All problems contain vehicle dependent fixed costs. However, variable costs are set to be one in all problems. In other words, total solution cost is the sum of fixed costs and total distance traveled.

Table 4.6 Data belonging to test instances

		A		B		C		D		E		F	
Test Instance	**Number of Nodes**	**Cap_v**	**$Cost_v$**	**Cap_v**	**$Cost_v$**	**Cap_v**	**$Cost_v$**	**Cap_v**	**$Cost_v$**	**Cap_v**	**$Cost_v$**	**Cap_v**	**$Cost_v$**
3	20	20	20	30	35	40	50	70	120	120	225		
4	20	60	1000	80	1500	150	3000						
5	20	20	20	30	35	40	50	70	120	120	225		
6	20	60	1000	80	1500	150	3000						
13	50	20	20	30	35	40	50	70	120	120	225	200	400
14	50	120	100	160	1500	300	3500						
15	50	50	100	100	250	160	450						
16	50	40	100	80	200	140	400						
17	75	50	25	120	80	200	150	350	320				
18	75	20	10	50	35	100	100	150	180	250	400	400	800

The test instances listed in Table 4.6 are solved using the proposed approach in Section 4.1. The solutions belonging to Test Instance 4 are given in the following sections. All other solutions can be found in the Appendix B.

4.2.1 Splitting into Subproblems

The threshold is set to be 36 at the first iteration. Then it is gradually increased by 2 units at each iteration. Model SC is solved to give two subsets at this threshold level. The iterations and the number of subproblems can be seen in Table 4.7.

Table 4.7 Threshold levels and the number of subproblems of Test Instance 4

Iteration	**Threshold**	**No. of subproblems**
1	36	2
2	38	2
3	40	1

As seen from Table 4.7, when threshold is increased to 40 units, the problem is not split. Rather, it stays as a whole. Therefore, the iterations stop at threshold level 40.

4.2.2 Vehicle Assignment

In the IFGP algorithm, which is employed in vehicle assignment step, firstly the two conflicting objectives (number of vehicles and fixed cost of vehicles given in Equations 4.3 and 4.4) are minimized with the system constraints of Model VA (Formulation 4.5). The solutions belonging to these two objectives can be seen in Table 4.8. In the vehicle assignment phase, the minimum number of vehicles and fixed cost turned out be 3 and 6000 respectively (Table 4.8). For both threshold levels, the same values are achieved.

Table 4.8 Minimum values of z_1 and z_2

	Number of Vehicles	**Fixed Cost of Vehicles**
Minimize z_1	3	7000
Minimize z_2	6	6000

Then the IFGP algorithm is operated. The solutions of the iterations can be seen in Table 4.9.

Table 4.9 Iterations of fuzzy goal programming phase belonging to Test Instance 4

Iteration No	**Number of Vehicles**	**Fixed Cost of Vehicles**	**Difference Between the Last Two Iterations**
0	3	7000	-
1	4	7000	0
2	5	6500	500
3	**6**	**6000**	**500**

The ***ExpectedLoss*** is calculated to be 33 (maximum of distance values to the depot node). As seen from Table 4.9, the difference in fixed cost between any of the two consecutive iterations is greater than the ***ExpectedLoss***. Therefore, the procedure stops when minimum fixed cost is reached (z_{2min}). The solution of the last iteration (Table 4.9) is selected. The subproblems can be seen in Figure 4.8 (The triangle in red denotes the depot node).

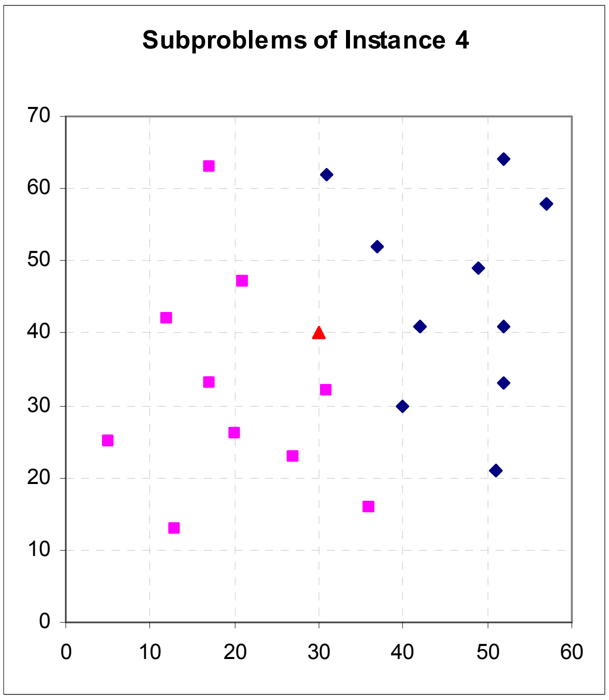


Figure 4.8 The two subproblems of Test Instance 4

4.2.3 Routing Phase

Each subproblem is solved by Model CPM (Formulation 4.7). The results at different threshold levels can be seen in Table 4.10. As expected, the best solution is achieved at threshold level 38 (higher threshold level). Therefore, this solution is the solution of Test Instance 4. The total traveling cost is 437,33; total fixed cost is 6000 (from Table 4.9) which makes total cost of 6437,33. The subproblem solutions at threshold level 38 can be seen separately in Table 4.11.

Table 4.10 Solutions belonging to Test Instance 4 at different threshold levels

Iteration	**Threshold**	**No. of Subproblems**	**No. of Vehicles**	**Fixed Cost**	**Traveling Cost**	**Total Cost**
1	36	2	6	6000	538,34	6538,34
2	38	2	6	6000	437,33	6437,33

Table 4.11 Solutions of subproblems belonging to Test Instance 4

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,…..,v) : M**	**Possible Combinations M^K**	**Fixed Cost**	**Traveling Cost**	**Computation Time (sec.)**
10	3	121	4	4^{121}	3000	212,43	**32**
10	3	121	4	4^{121}	3000	224,90	13

The subproblems are solved in parallel. Therefore, the computation time of the complete problem is the maximum of the computation times of subproblems. The computation time of Test Instance 4 is 32 seconds (max[13 ; 32]).

It is known that as the number of variables and their allowable values increase, the complexity of the problem increases. The complexity of the subproblems belonging to Test Instance 4 is 4^{121}.

Some of the alternative combinations are infeasible. Among the feasible ones, complete enumeration is made by OPL Script algorithm and the optimum solution to each subproblem is found. The routes of Test Instance 4 can be seen in Figure 4.9.

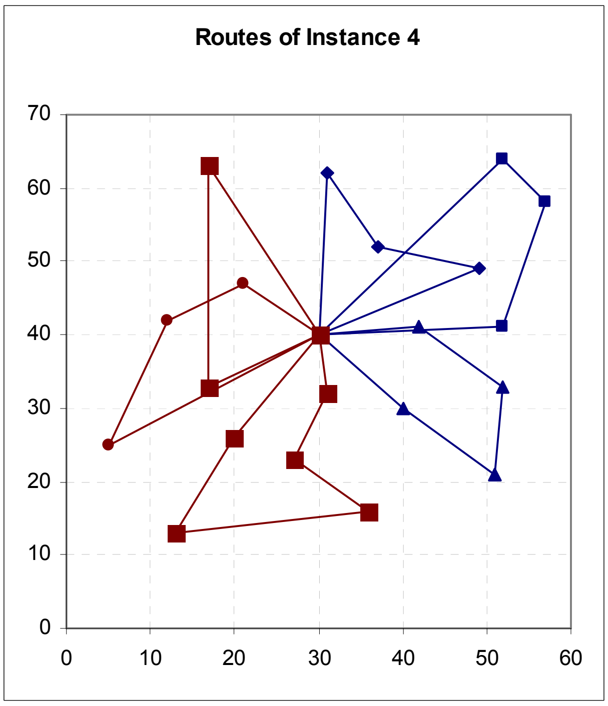


Figure 4.9 Routes of Instance 4. The routes belonging to the two subproblems are shown in different colors.

All test instances from Gendreau et. al. (1999) are solved. Detailed solutions can be seen in Appendix B.

4.3 Computational Results

All test instances are solved and the results are compared with the best known solutions in the literature (Table 4.12).

It can be seen from Table 4.12 that the Threshold Algorithm is able to find the best known solutions in the literature for some of the instances (Instance 4 & 6). For others the solution cost turned out to be within at most 3% deviation from the best known. On the other hand, solution times of the threshold algorithm and the literature solutions cannot be compared. This is due to the fact that the computer systems employed in the procedures are completely different. However, it can be said that solution times of the proposed procedure are quite reasonable. So, the threshold algorithm provides solutions in practically usable solution times.

Table 4.12 Comparison of Threshold algorithm and the best known solutions in the literature

Problem No	**Number of Nodes**	**Taillard 1999**	**Gendreau et. al. 1999**	**Choi and Tcha 2007**	**Threshold Algorithm**		
					Cost	**Deviation in Cost**	**Solution Time**
3	20	961,03	961,03	961,03	983,1	2,24%	52
4	**20**	**6437,3**	**6437,33**	**6437,3**	**6437,33**	**0,00%**	**32**
5	20	1008,59	1007,05	1007,05	1052,15	4,29%	102
6	**20**	**6516,5**	**6516,47**	**6516,5**	**6516,47**	**0,00%**	**131**
13	50	2413,78	2408,41	2406,36	2440,78	1,41%	113
14	50	9119,03	9119,03	9119,03	9138,25	0,21%	374
15	50	2586,37	2586,37	2586,37	2620,54	1,30%	200
16	50	2741,5	2741,5	2720,43	2746,30	0,94%	243
17	75	1747,24	1749,5	1744,83	1783,33	2,16%	340
18	75	2373,63	2381,43	2371,49	2394,16	0,95%	442
19	100	8661,81	8675,16	8664,29	8734,35	0,83%	453
20	100	4047,55	4086,76	4039,49	4139,80	2,42%	261

Also, it should be noticed that the deviation in solution cost is independent from the number of nodes in the problem. In addition, as the number of nodes increases the computation time of the Threshold Algorithm stays same on the average. This is due to the clustering phase of the algorithm. The number of clusters increase but the size of the clusters are approximately equivalent. Therefore the computation time does not increase noticeably.

However, the approaches presented by Taillard 1999, Gendreau et al. 1999 and Choi and Tcha (2007) are global and hence computation time increases quadratically as the size of the problem increases. In other words, these approaches handle the full size problem without splitting into subproblems. So, as the problem size increases, computation times of these approaches would increase much more. Therefore it is expected that, for larger scale problems, the Threshold Algorithm will provide good solutions within competitive time compared to global approaches. In conclusion, it can be said that for real life cases where there are several hundreds of demand points to be handled, Threshold Algorithm may be functional to apply.

4.4 Application of the Proposed Algorithm To Real Life Case: Fresh Goods Distribution of a Retail Store

The real life fresh goods distribution problem of a retail chain store is handled with the Threshold Algorithm in this section (Mizrak Ozfirat, Ozkarahan, 2006).

4.4.1 Problem Definition

In this study the distribution of fresh meat from a central depot to retail stores is handled. The depot belongs to a retail chain store located in Izmir Turkey. There are 41 demand points to which the depot should deliver meat weekly (Figure 4.10).

The distances between demand points are in kilometers and measured from main roads. In other words, since there exist physical road and land restrictions, some of the distance figures may not satisfy triangle inequality. The firm owns nine vehicles assigned for this distribution. The data belonging to vehicles are given in Table 4.13.

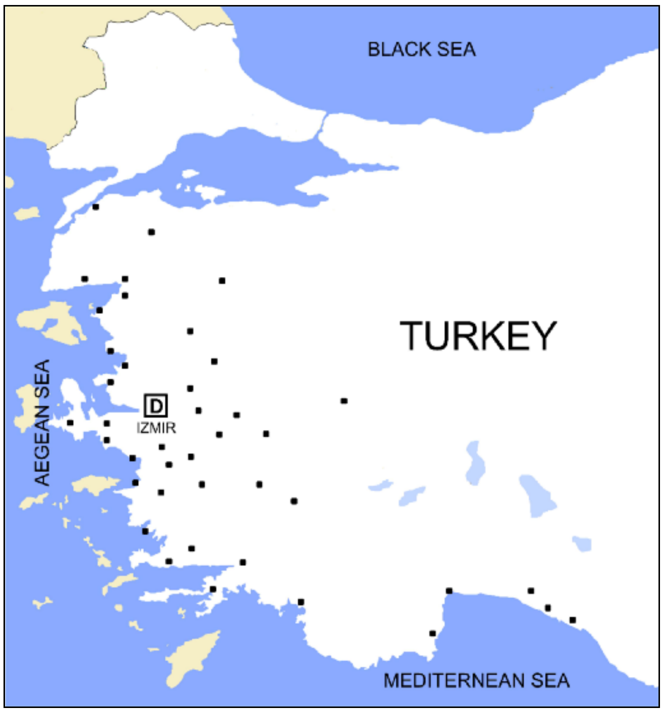


Figure 4.10 Demand points of the real life distribution problem

Table 4.13 Data belonging to real life distribution problem

Vehicle	Capacity	Fixed Cost	Vehicle	Capacity	Fixed Cost
1	11484	1148,4	6	11824	1182,4
2	11144	1114,4	7	10514	1051,4
3	14314	1431,4	8	15724	1572,4
4	10514	1051,4	9	12000	1200
5	9980	998			

4.4.2 Splitting the Problem into Subproblems

The problem is designed as a fixed fleet heterogeneous vehicle routing problem. The first threshold level is taken to be 80km. When set covering model is solved, 13 sets are selected. This means that at least 13 vehicles are required. In this case the solution is infeasible (since there exist 9 vehicles). Therefore threshold is increased to 100km. and after that it is increased by 20 km at each iteration.

The threshold levels at each iteration and the number of subproblems found by Model SC (Formulation 4.1) can be seen in Table 4.12. At the first three iterations, the solution is infeasible since the number of subproblems exceed the number of vehicles. The first feasible solution is achieved at the 4th iteration when threshold level is set to 140km. There exist 6 subproblems at this threshold level. So, at east 6 vehicles are necessary. In the next step of the procedure, vehicles are assigned to subproblems.

Table 4.14 Solutions of Model SC for real life case

Iteration	**Threshold**	**No. of subproblems**	**Min. No. of Vehicles Necessary**
1	80	13	**13**
2	100	12	**12**
3	120	10	**10**
4	140	6	6
5	160	6	6
6	180	5	5
7	200	4	4
8	220	4	4
9	240	4	4

4.4.3 Vehicle Assignment to Subproblems

At this step, as explained previously, there exist two conflicting objectives. These are number of vehicles and fixed cost of vehicles respectively. In the IFGP algorithm, which is employed in vehicle assignment step, firstly the two objectives (given in Equation 4.3 and 4.4) are minimized with the system constraints of Model VA (Formulation 4.5). The solutions belonging to these two objectives can be seen in Table 4.15.

Table 4.15 Solutions of Model VA when z_1 and z_2 are minimized

Iteration	**Threshold Level**	**Objective Function**	**Number of Vehicles**	**Fixed Cost of Vehicles**
4	140	Minimize z_1	6	7010
		Minimize z_2	6	6544
5	160	Minimize z_1	6	7435
		Minimize z_2	6	6793
6	180	Minimize z_1	5	6153
		Minimize z_2	5	5645
7	200	Minimize z_1	4	5299
		Minimize z_2	4	5052
8	220	Minimize z_1	4	5102
		Minimize z_2	4	4769
9	240	Minimize z_1	4	5236
		Minimize z_2	4	4311

Since the first feasible solution can be achieved at threshold level 140, the first vehicle assignment takes place at this iteration. As seen from Table 4.15, the two objectives are not conflicting in this case. In other words, the number of vehicles turns out to be equal when both objectives are minimized. Therefore, at threshold level 140, the solution in Table 4.15, which minimizes the vehicle fixed costs, is selected. Similarly, at all threshold levels, the solution which minimizes vehicle fixed costs are selected. The subproblems belonging to iteration 9 (threshold level 240) can be seen in Figure 4.11. The subproblems belonging to all other iterations are in Appendix C.

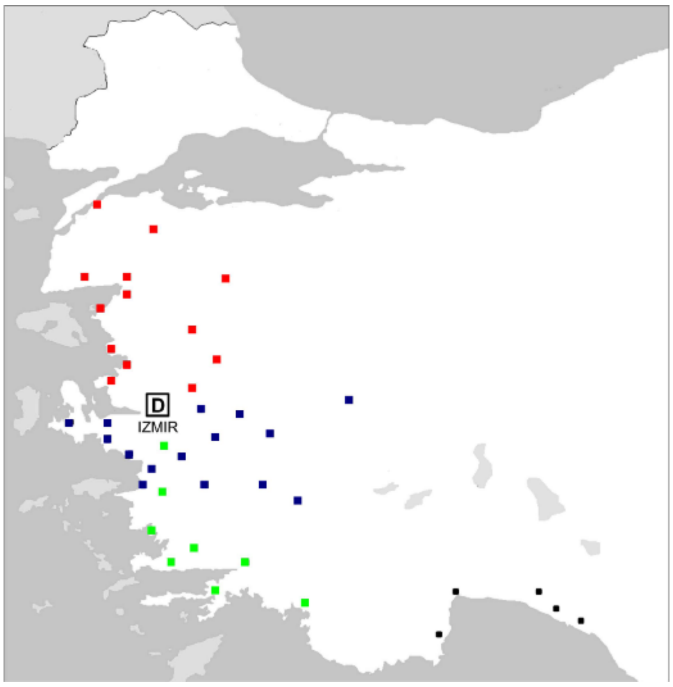


Figure 4.11 Subproblems at threshold level 240 km

4.4.4 Routing Phase

All subproblems are solved using Model CPM (Formulation 4.7) together with the OPL Script (ILOG, 2003) SEA. The solutions of all iterations can be seen in Table 4.16.

Table 4.16 Solutions of Model CPM for real life case.

Iteration	**Threshold**	**No. of Subproblems**	**No. of Vehicles**	**Fixed Cost**	**Traveling Cost**	**Total Cost**
1	80	13	**Infeasible**			
2	100	12				
3	120	10				
4	140	6	6	6544	4659	11203
5	160	6	6	6793	4853	11646
6	180	5	5	5645	4774	10419
7	200	4	4	5052	4369	9421
8	220	4	4	4769	4072	8841
9	***240***	***4***	***4***	***4311***	***3792***	***8103***

When threshold level is set to 260km, one of the subproblems turn out to be insolvable. Therefore, the iterations stop at threshold level 240. The solution of this iteration is the best solution achieved in the procedure. The routes belonging to this threshold level can be seen in Figure 4.12 and Table 4.17.

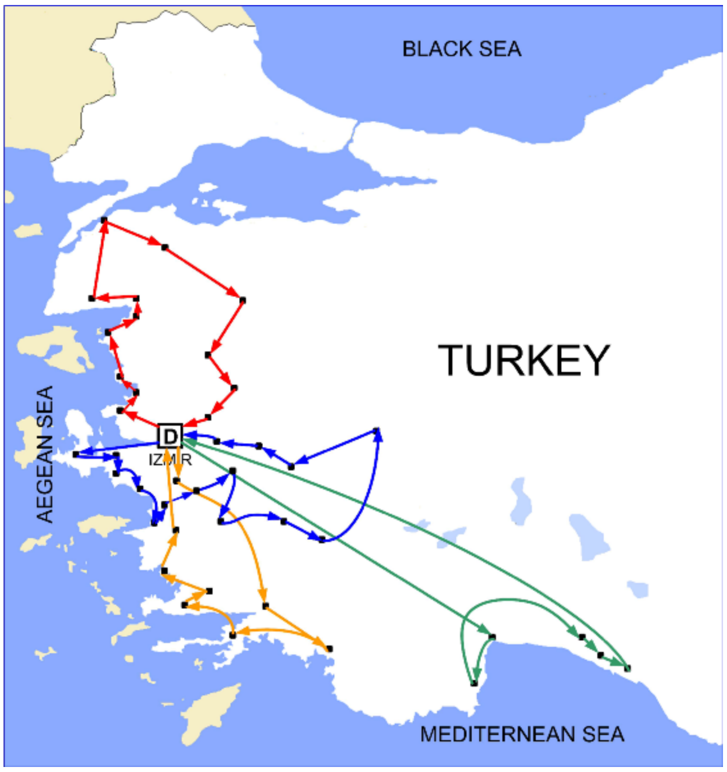


Figure 4.12 Routes of real-life distribution problem

Table 4.17 Final solution of real life distribution problem (Total solution cost=8103)

Vehicle	Distance Traveled	Solution Time (sec.)	Route
2	1286	1	0-6-22-24-7-2-0
5	767	243	0-4-21-18-9-12-19-5-14-13-10-35-1-25-0
3	816	2	0-38-28-20-26-11-27-17-36-0
4	923	33	0-15-40-33-31-23-34-37-30-8-29-16-3-3-32-39-0
Overall	***3792***	***243***	

As seen from Table 4.17, total traveling cost is 3792 and total cost is 8103. In the current operation of the firm, 5 vehicles are employed with total fixed cost of 5362. Total cost of the current performance of the firm is 3902. The comparison between the current performance of the firm and the threshold algorithm can be seen in Table 4.18.

Table 4.18 Comparison of current performance of the firm and the threshold algorithm

	No. of Vehicles Allocated	**Fixed Cost (YTL)**	**Traveling Cost (YTL)**	**Total Cost**	**Solution Time**	**Improvement (%)**
Current Performance of the firm	5	5362	3908	9270	-	-
Threshold Algorithm	4	4311	3792	8103	243	12,59

In conclusion, the proposed algorithm in this research provided improvement for the fresh goods distribution of the retail store in terms of both fixed cost and traveling cost. The new solution offered to the retail store provides 12,59% overall improvement. It can be said that the decrease in fixed cost is mainly provided by decreasing the number of vehicles. Also, traveling cost is decreased by changing the routes of vehicles by help of the CP model.

CHAPTER FIVE
THRESHOLD ALGORITHM FOR SPLIT DELIVERY VEHICLE ROUTING PROBLEM

In the classical vehicle routing problem (VRP) a fleet of homogeneous vehicles is available to serve a set of customers with known demand. Each customer is required to be visited by exactly one vehicle and the objective is to minimize the total distance traveled. In the split delivery vehicle routing problem (SDVRP), introduced by Dror and Trudeau (1989), the restriction that each customer has to be visited exactly once is removed, i.e., split deliveries are allowed. Therefore, SDVRP can be considered as a relaxation of the capacitated VRP (CVRP). An example distribution plan of CVRP and SDVRP can be seen in Figure 5.1.

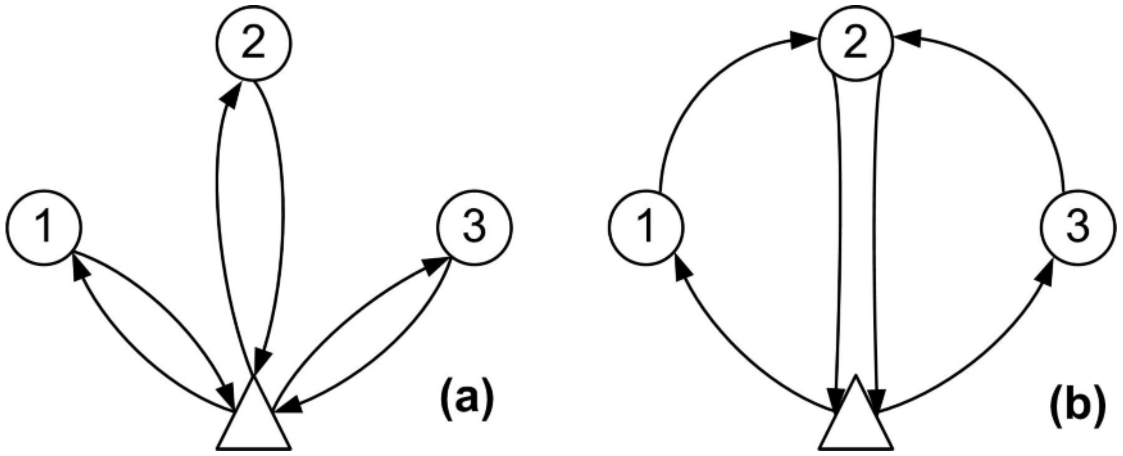


Figure 5.1 Example of a (a) CVRP distribution plan (b) SDVRP distribution plan. The triangle represents the depot node.

Recently, there has been an increase in interest for the SDVRP. It is a challenging variant of the vehicle routing problem with significant potential for practical implications. Since SDVRP is a relaxation of CVRP, it is expected to provide decreased delivery costs. Recently, Archetti et al. (2006) have shown that SDVRP provides delivery costs less than or equal to CVRP. In addition, the authors have proved that the reduction in delivery costs will be at most 50% (Equation 5.1).

$$\frac{DistributionCosts_{CVRP}}{2} \leq DistributionCosts_{SDVRP} \leq DistributionCosts_{CVRP} \quad (5.1)$$

As seen from Equation 5.1, the cost of an optimal solution when split deliveries are allowed is always greater than half the cost of an optimal solution when splitting of deliveries is not allowed. However, there are also some disadvantages of this system such as higher customer inconvenience, more complex administration and accounting. Therefore, companies need to carefully evaluate these trade-offs.

SDVRP can be divided into two main groups where

- split deliveries are inevitable where demands are larger than the vehicle capacity.
- split deliveries are allowed where demands are smaller than the vehicle capacity.

In the first case, there is no alternative than splitting the deliveries. The rear case is the one which is referred generally as SDVRP in the literature. And hence, it is the subject of this research.

In the first section of this chapter, the Threshold Algorithm, which is proposed in Chapter 4 for HVRP, is modified according to split deliveries. Section 2 gives the performance tests on SDVRP literature problems.

In section 3, the Threshold Algorithm for HVRP (from Chapter 4) and SDVRP (from Section 5.1) are combined to handle the VRP both considering a heterogeneous fleet and split deliveries. That is HVRP with split deliveries is considered. To the best of our knowledge, this dissertation is the first research which considers HVRP with split deliveries. In the next section, HVRP test instances from the literature are solved under split delivery assumption. Since there are no studies in the literature considering HVRP with split deliveries, these test instances are important in the sense to provide some benchmark values for the literature.

Finally, the Threshold Algorithm proposed for HVRP with split deliveries is applied to the fresh goods distribution problem of the retail chain store discussed in Chapter 4. The two distribution strategies, which are non-split and split delivery strategies respectively, are compared in terms of solution cost and solution time. A distribution strategy based on the findings is offered to the firm.

5.1 Modified Threshold Algorithm for Split Delivery Vehicle Routing Problem

The Threshold Algorithm developed for HVRP in Chapter 4 is modified according to SDVRP. The flow of the modified Threshold Algorithm for SDVRP can be seen in Figure 5.2 (Mizrak Ozfirat, Ozkarahan, 2007).

The main differences in between the two algorithms lie in the vehicle assignment and routing phases. Basically, due to the characteristics of SDVRP, integer programming (IP) is employed in the vehicle assignment phase. In the routing phase of the algorithm, firstly constraint programming (CP) approach is employed. However, due to some inefficiencies of the CP approach, an IP based approach is developed and applied to SDVRP.

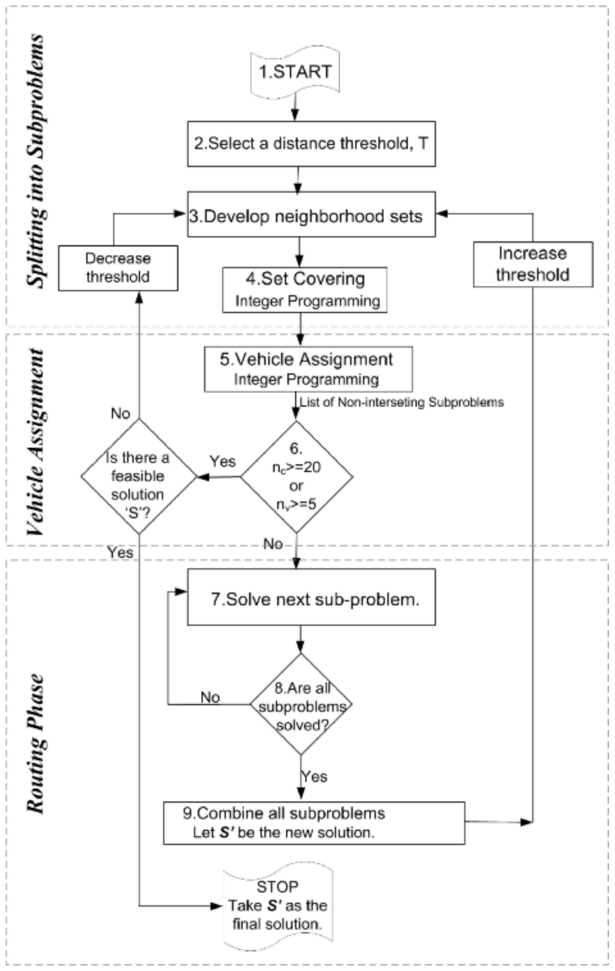

Splitting into Subproblems
Vehicle Assignment
Routing Phase
1.START
2.Select a distance threshold, T
3.Develop neighborhood sets
Decrease threshold
4.Set Covering Integer Programming
Increase threshold
5.Vehicle Assignment Integer Programming
List of Non-interseting Subproblems
No
Is there a feasible solution 'S'?
Yes
6. n_c>=20 or n_v>=5
Yes
No
7.Solve next sub-problem.
No
8.Are all subproblems solved?
Yes
9.Combine all subproblems Let S' be the new solution.
STOP Take S' as the final solution.

Figure 5.2 Modified Threshold Algorithm for SDVRP

5.1.1 Splitting the problem

The beginning steps of the algorithm are same as the Threshold Algorithm developed for HVRP (in Chapter 4).

Step 1: Start

Step 2: Select a threshold ***T***. Let there be ***"n_c"*** customers to be served.

Step 3: Develop the neighborhood sets of each customer within distance ***T*** to the customer.

Step 4: Set Covering: The same model as in Model SC of Chapter 4 (Formulation 4.1, page 76) is employed. Similar to HVRP, minimum number of sets are selected in the most centralized way.

5.1.2 Vehicle Assignment

For HVRPs, there exists a conflict between the number of vehicles and the fixed cost of vehicles at vehicle assignment step of the algorithm. In order to deal with this conflict an interactive fuzzy goal programming (IFGP) procedure is developed and used. However, for SDVRPs, the vehicle fleet is assumed to be homogeneous. That is all vehicles are identical in capacity and fixed cost. There is no conflict between the number of vehicles and their fixed costs. As the number of vehicles selected increases, fixed cost increases directly proportional to the number. Therefore, there is no need for the IFGP approach. Instead, an IP model, which minimizes the total number of vehicles selected, is employed for the vehicle assignment step of SDVRP.

Step 5: The IP model, called SplitDeliveryVehicleAssignment (Model SDVA), is given in Formulation 5.2 and the notation used is given in Table 5.1.

Table 5.1 Notation used in vehicle assignment model for SDVRP

$Parameters:$

$n_c : Number\ of\ nodes.\ \ n_v : Number\ of\ vehicles.$

$Demand_j\!: Demand\ of\ node\ j,\ j : 0..n_c.$

$Cap_v : Capacity\ of\ vehicle\ v, v : 1..n_v.$

$Sets:$

$Nodes = \{1,2,.....,n_c\}$

$T^* = \{t_i^* : i \in Nodes\} : Optimum\ solution\ of\ the\ set\ covering\ model\ (Formulation\ 4.1).$

$Subproblems = \{i \in Nodes : t_i^* = 1\}.$

$Vehicles = \{1,2,.....,n_v\}$

$Decision\ \ Variables:$

$$y_{ij} : \begin{cases} 1\ if\ customer\ j\ is\ assigned\ to\ subproblem\ i \\ 0\ if\ customer\ j\ is\ not\ included\ in\ subproblem\ i \end{cases} ,i \in Subproblems\ ,j \in Nodes.$$

$$w_{iv} : \begin{cases} 1\ if\ vehicle\ v\ is\ assigned\ to\ subproblem\ i \\ 0\ otherwise \end{cases} ,i \in Subproblems\ ,v{:}1..n_v.$$

$$
\begin{array}{lll}
\textit{Model SDVA}: & & \\
\textit{Min.} \quad \sum_{i}\sum_{v} w_{iv} & & (5.2a) \\
\textit{subject to} & & \\
\sum_{i \in Subproblems} y_{ij} = 1 & \forall j \in Nodes & (5.2b) \\
y_{ij} \leq M_{ij} & \forall i \in Subproblems, j \in Nodes & (5.2c) \\
\sum_{v \in Vehicles} Cap_v w_{iv} \geq \sum_{j \in Nodes} y_{ij} Demand_j & \forall i \in Subproblems & (5.2d) \\
\sum_{i \in Subproblems} w_{iv} \leq 1 & \forall v \in Vehicles & (5.2e) \\
w_{iv} \in \{0,1\} & \forall i \in Subproblems, \forall v \in Vehicles & (5.2f) \\
y_{ij} \in \{0,1\} & \forall i \in Subproblems, \forall j \in Nodes & (5.2g)
\end{array}
\qquad (5.2)
$$

In Formulation 5.2, the objective function ***(5.2a)*** minimizes the total number of vehicles selected. Constraints ***(5.2b)*** state that all nodes should appear in exactly one of the subproblems and constraints ***(5.2c)*** state each node must be assigned to a subproblem within its neighborhood. Capacity restrictions are included in the model with constraints ***(5.2d)***. Constraint set ***(5.2e)*** assures that each vehicle is used only once. Finally, binary constraints are given by ***(5.2f)*** and ***(5.2g)***.

It should be realized that Model SDVA is derived from Model VA of HVRP (Chapter 4, Formulation 4.5, page 81). There are two objectives in HVRP. These are to minimize total number of vehicles and to minimize total fixed cost respectively. In SDVRP, since these two bring out the same solution, the objective function is set to minimize total number of vehicles (Equation 5.2a). In addition, constraints ***(5.2b)*** to ***(5.2g)*** are the system constraints of Model VA. In other words, Model SDVA is achieved by turning the fuzzy goal programming model (Model VA) of HVRP into an IP model.

Once this model is solved, the subproblems and the vehicles assigned to each subproblem are obtained.

Step 6: Similar to HVRP, the conditions to stop the algorithm are determined to be $n_c >= 20$ (number of nodes) or $n_v >= 5$ (number of vehicles).

5.1.3 Routing Phase

In the routing phase of the Threshold Algorithm for HVRP, CP is employed. In this section, firstly, this model is modified according to split delivery assumptions. However, this model turned out to be unsuccessful for SDVRP. Hence, an IP model is also built for the routing phase. Both CP and IP approaches are explained in this section.

5.1.3.1 Routing by Constraint Programming

Step 7: In SDVRP, the delivery of a customer can be split between two or more vehicles. Therefore, the model (Model CPM given in Formulation 4.7 of Chapter 4, page 87) which finds out the routes of the vehicles should be revised considering this assumption. Firstly, another decision variable is necessary:

$$A_{iv} : \textit{Capacity of vehicle v allocated to node i.}$$

The new routing model, called the SplitDeliveryModel (Model SDM) is given in Formulation 5.3. The notation used is same as in Model CPM (Table 4.5, page 87).

$$
\begin{array}{lll}
\textit{Model SDM}: & & \\
\textit{Solve} & & \\
\sum_{j \in Total_k} (X_{ij} = v) \le 1 & \forall i \in Nodes_k,\ v \in Vehicles & (5.3a) \\
\sum_{i \in Total_k} (X_{ij} = v) \le 1 & \forall j \in Nodes_k,\ v \in Vehicles & (5.3b) \\
\sum_{i \in Total_k} (X_{i0} = v) = 1 & \forall v \in Vehicles & (5.3c) \\
\sum_{j \in Total_k} (X_{0j} = v) = 1 & \forall v \in Vehicles & (5.3d) \\
\sum_{j \in Total_k} X_{ij} = \sum_{j \in Total_k} X_{ji} & \forall i \in Total_k & (5.3e) \\
\sum_{i \in Nodes_k} A_{iv} \le Cap_v & \forall v \in Vehicles & (5.3f) \\
\sum_{v \in Vehicles} A_{iv} = Demand_i & \forall i \in Nodes_k & (5.3g) \\
A_{iv} \le B \sum_{j \in Total_k} (X_{ij} = v) & \forall i \in Nodes_k,\ v \in Vehicles & (5.3h) \\
A_{iv} \ge \sum_{j \in Total_k} (X_{ij} = v) & \forall i \in Nodes_k,\ v \in Vehicles & (5.3i)
\end{array}
\qquad \textbf{(5.3)}
$$

Constraints ***(5.3a)*** and ***(5.3b)*** allow split deliveries. In addition, each vehicle should visit the depot once since the vehicles assigned to the problem are already known from the vehicle assignment step. This is included in the model with constraints ***(5.3c)*** and ***(5.3d)***. Constraint set ***(5.3e)*** provides that a vehicle visiting a customer should leave that customer. Capacity and demand requirements are stated using the decision variable ***A*** in ***(5.3f)*** and ***(5.3g)***. That is, constraints ***(5.3f)*** state that the goods carried in a vehicle should be less than or equal to its capacity and ***(5.3g)*** state that the goods carried in all vehicles for a node must be equal to its demand. Finally, the relationship between variables ***A*** and ***X*** are built by constraints ***(5.3h)*** and ***(5.3i)***. In constraints ***(5.3h)***, ***B*** represents a big number. It is stated that if vehicle ***v*** does not visit node ***i*** then, the capacity of vehicle ***v*** allocated to node ***i*** should be 0 (Constraints ***5.3h***). On the other hand, if vehicle ***v*** leaves node ***i***, then the capacity of vehicle ***v*** allocated to node ***i*** should be a positive number (Constraints ***5.3i***). The model is written and solved in OPL Studio 3.7 (ILOG, 2003). The code of the model is given in Appendix D.

Similar to HVRP formulation, subtour elimination constraints are also missing in this model in order to decrease computation time. Subtour Elimination Algorithm (SEA) which is defined in Chapter 4 (page 89) is modified according to split delivery assumption and employed for SDVRP. Shortly, it can be said that SEA makes complete enumeration among the feasible solutions of Model SDM and finds the minimum distance solution which does not contain any subtours. The flow of SEA can be seen in the following. The OPL code of SEA is given in Appendix D.

OPL Script SEA for Model SDM:

```
1. Start;
2. MinimumDistance:=MaximumInteger;
3. Let $S$ be the next solution of model SDM;
4. If $S \neq \emptyset$ then 5, else 9;
5. Distance:=Total Distance Traveled in $S$;
6. If S is a full solution (does not contain sub-tours), then 7, else 8
7. If Distance<MinimumDistance, then MinimumDistance:=Distance;
8. Go to 3;
9. Stop;
```

Allowing split deliveries in a VRP relaxes the constraint that all nodes should be visited exactly once. Therefore, the search space enlarges in a considerable way. When the CP procedure is employed to solve the SDVRP benchmark instances in the literature, it is seen that the problems with three or more vehicles are insolvable. Therefore, the procedure failed to provide any solution for any one of the benchmark problems. Since CP approach turned out to be unsuccessful, an IP approach is developed for SDVRP.

5.1.3.2 Routing by Integer Programming

Revised Step 7: Model SDM, which is a CP model, is replaced with an IP model to find the routes of the vehicles. The new model is referred as IntegerProgrammingModel (Model IPM) and given in Formulation 5.4. The notation used is given in Table 5.2. The OPL code of Model IPM is given in Appendix D.

Table 5.2 Notation used in Model IPM

Parameters :

n_c : *Number of nodes to be served.* n_v : *Number of vehicles.*

D_{ij} : *Distance from node i to node j.* $Demand_j$: *Demand of node j.*

Cap_v : *Capacity of vehicle v.*

Noofsubtours: No of subtours identified by the OPL Script Algorithm (In IP approach of routing phase).

$subtours_{ti}$: *The* i^{th} *node which is included in subtour* $t, t : 1..Noofsubtours$.

rhs_t*:Maximum allowable value to break subtour t, t:*1..*Noofsubtours.* B : *A big number.*

Sets :

$Nodes = \{1,2,.....,n_c\}$

$Y^* = \{y^*_{ij} : i \in Subproblems, j \in Nodes\}$*:Optimum solution of Model VA.*

$Nodes_k = \{j \in Nodes : y^*_{kj} = 1\} \quad \forall k \in Subproblems; Let \ |Nodes_k| = a_k \quad \forall k \in Subproblems$

$Total_k = Nodes_k \cup \{0\}$;"0" *denotes depot node.*

$Subtourrange = \{1,2,...,Noofsubtours\}$

$Vehicles = \{1,...,n_v\}$

Decision Variables : *Routing Phase*

$$X_{ijv} : \begin{cases} 1 & \text{If vehicle } v \text{ travels from node } i \text{ to node } j. \\ 0 & \text{Otherwise.} \end{cases} \qquad ,i : 0..n_c, j : 0..n_c, v : 1..n_v.$$

A_{iv} : *Capacity of vehicle v allocated to node i.*

$$
\begin{aligned}
&\textit{Model IPM}: \\
&\textit{Minimize} \sum_{i \in Total_k} \sum_{j \in Total_k} \sum_{v \in Vehicles} X_{ijv} * D_{ij} && (5.4a) \\
&\textit{Solve} \\
&\sum_{j \in Total_k} X_{ijv} \le 1 \qquad \forall i \in Nodes_k, \forall v \in Vehicles && (5.4b) \\
&\sum_{i \in Total_k} X_{ijv} \le 1 \qquad \forall j \in Nodes_k, \forall v \in Vehicles && (5.4c) \\
&\sum_{i \in Total_k} X_{i0v} = 1 \qquad \forall v \in Vehicles && (5.4d) \\
&\sum_{j \in Total_k} X_{0jv} = 1 \qquad \forall v \in Vehicles && (5.4e) \\
&\sum_{i \in Total_k} X_{ikv} = \sum_{j \in Total} X_{kjv} \qquad \forall k \in Total_k, \forall v \in Vehicles && (5.4f) \\
&\sum_{i \in Nodes_k} A_{iv} \le Cap_v \qquad \forall v \in Vehicles && (5.4g) \\
&\sum_{v \in Vehicles} A_{iv} = Demand_i \qquad \forall i \in Nodes_k && (5.4h) \\
&A_{iv} \le B * \sum_{j \in Total} X_{ijv} \qquad \forall i \in Nodes_k, \forall v \in Vehicles && (5.4i) \\
&A_{iv} \ge \sum_{j \in Total_k} X_{ijv} \qquad \forall i \in Nodes_k, \forall v \in Vehicles && (5.4j) \\
&\sum_{i \in [1..a_k]} X_{subtours_{ti} subtours_{ti+1} v} \le rhs_t \qquad \forall t \in Subtourrange, \forall v \in Vehicles && (5.4k) \\
&\sum_{i \in [1..a_k]} X_{subtours_{ti+1} subtours_{ti} v} \le rhs_t \qquad \forall t \in Subtourrange, \forall v \in Vehicles && (5.4l) \\
&X_{ijv} \in \{0,1\} \qquad \forall i,j \in Nodes_k, \forall v \in Vehicles && (5.4m)
\end{aligned}
\qquad \textbf{(5.4)}
$$

The objective function ***(5.4a)***, minimizes total distance traveled. Constraints ***(5.4b)*** and ***(5.4c)*** state that each customer can be visited by one or more vehicles (split deliveries). In addition, each vehicle should visit the depot once. This is included in the model with constraints ***(5.4d)*** and ***(5.4e)***. Constraint set ***(5.4f)*** provides that a vehicle visiting a customer should leave that customer. Constraints ***(5.4g)*** state that the goods carried in a vehicle should be less than or equal to its capacity and ***(5.4h)*** state that the goods carried in all vehicles for a node must be equal to its demand. Constraints ***(5.4i)*** and ***(5.4j)*** build the relationship between variable ***A*** and ***X***. Constraints ***(5.4k)*** and ***(5.4l)*** are subtour elimination constraints. However, it is not possible to include all subtour constraints into the IP model.

When all subtour constraints are included, the model becomes insolvably large. Therefore, an effective procedure to handle the subtour constraints is necessary. However, this procedure cannot be similar to the Subtour Elimination Algorithm of Model SDM. One difference between CP and IP is that in a CP model, all alternative optimum solutions can be found. However, in an IP model only one optimal solution is provided. Therefore, it is not possible to make complete enumeration between all optimal solutions of an IP model (since only one solution would be provided). Previously, in SEA (of Model SDM), the minimum cost routes which do not contain any subtours are found through complete enumeration on the solutions of the CP model. But, since we cannot make complete enumeration on the solutions of Model IPM, a new subtour elimination algorithm (SEA for Integer Programming) is developed for Model IPM. The new algorithm is also written in OPL Script. The algorithm looks at the solution of Model IPM, finds the subtours (if there are) and adds constraints back into Model IPM in order to avoid these subtours (Constraints ***5.4k*** and ***5.4l***). The flow of SEA for Integer Programming is given in the following:

1. Start;
2. Solve Model IPM, let ***S*** be the solution;
3. If ***S*** does not contain any subtours then go to 7;
4. Identify subtours;
5. Add corresponding subtour elimination constraints to Model IPM (Constraints 5.4k & 5.4l);
6. Go to 2;
7. ***S*** is the optimum solution;

The connection between the SEA Algorithm for Integer Programming and Model IPM is built by constraints 5.4k and 5.4l. The ***subtours*** matrix (employed in 5.3k and 5.3l) contains a subtour in each of its rows (eg. 1-2-3-1). The matrix is empty at the very first iteration. That is, Model IPM is solved without any subtour constraints at the beginning. Then OPL Script algorithm takes this solution, identifies the subtours and adds them to the ***subtours*** matrix. Model IPM is resolved with these subtour constraints (***5.4k*** and ***5.4l***). The procedure goes on iteratively until no subtours exist in the solution, ***S***. Then, ***S*** becomes the optimum. The OPL code of SEA for Integer Programming is given in Appendix D.

The finalization steps of the algorithm are similar to the threshold algorithm developed for HVRP (in Chapter 4 of this dissertation).

Step 8: If all subproblems are finished then go to Step 9, else go to Step 7.

Step 9: Combine the solutions of all subproblems to give the solution of the main problem.

Step 10: Increase threshold by **'*s*'** units, T:=T+***s***. Go to Step 3.

5.1.3.3 Comparison of CP and IP in the Routing Phase

In the routing phase of the Threshold Algorithm developed for HVRP, CP is employed. All subproblems are solved to optimum by the CP approach (from Section 4.1.3, page 86). However, when solving SDVRP, CP approach failed to provide any solution to problems with two or more vehicles. Therefore, an IP approach is developed for the routing phase of SDVRP problems. Test problems from the literature are successfully solved by the IP approach (Performance tests are given in Section 5.2).

In summary, two different models (IP and CP) are proposed for two different problems, HVRP and SDVRP. This is shown in the following:

i. CP employed for HVRP: Subproblems are solved successfully to optimum.
ii. CP employed for SDVRP: No solution could be achieved.
iii. IP employed for HVRP: Since CP is successful for HVRP, there is no need to solve the same problems with IP. Only comparison on computation time is necessary.
iv. IP employed for SDVRP: Subproblems are solved successfully to optimum.

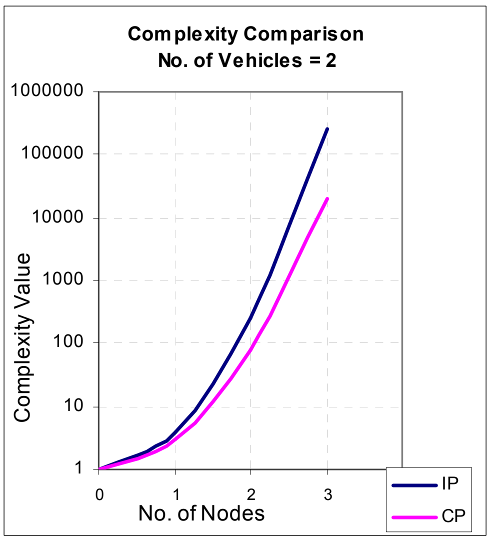
Looking at the case (i) and (iii) given above, it is clear that a comparison between CP and IP is necessary. It is known that both approaches are able to achieve the optimum solutions. Therefore, a comparison on solution time is necessary. Hence, a complexity analysis is done between IP and CP models both for HVRP and SDVRP.

- Complexity analysis for HVRP: The analysis on IP and CP models can be seen in Table 5.3. It is assumed that there exist "***n***" nodes to be served.

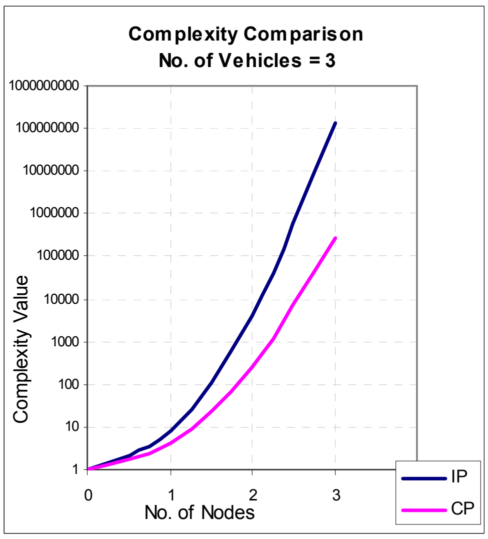
Table 5.3 Comparison on the complexities of IP and CP approaches on HVRP

	IP			**CP**		
No.of Vehicles	**No.of Variables**	**No.of Available Values for Each Variable**	**Complexity Function**	**No.of Variables**	**No.of Available Values for Each Variable**	**Complexity Function**
1	n^2	2	2^{n^2}	n^2	2	2^{n^2}
2	$2n^2$	2	2^{2n^2}	n^2	3	3^{n^2}
3	$3n^2$	2	2^{3n^2}	n^2	4	4^{n^2}
4	$4n^2$	2	2^{4n^2}	n^2	5	5^{n^2}

It can be seen from the table that, for a single vehicle case, the complexity functions are similar. For two or more vehicles, complexity function of IP takes values greater than the complexity function of CP for any value of "***n***". To see this more clearly, the complexity functions are graphed in Figure 5.3.



(a)



(b)

Figure 5.3 Complexity functions of IP and CP when no. of vehicles is (a) 2 and (b) 3

According to Figure 5.3, it can be stated that complexity of IP model is larger than CP model for all $n>0$. Also, it is seen from the graph that as n or the number of vehicles increases, the difference between complexities increases quadratically. Therefore, it is expected that solution time of IP approach would be much larger than CP approach in HVRP problems.

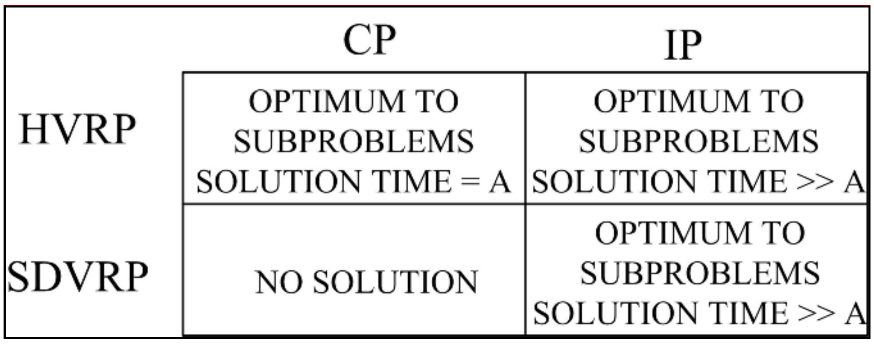
- Complexity analysis for SDVRP: Similarly, we need to compare the two approaches for SDVRP problems. In Table 5.4, the complexity analysis between IP and CP on SDVRP problems can be seen. It is assumed that there exist "n" nodes to be served, and "d_i" is the demand of node i. It should be noted that in SDVRP models, in addition to the variable (X), which determines routes, there exists another variable (A), which determines the amount of capacity of each vehicle allocated to each node.

Table 5.4 Comparison on the complexities of IP and CP approaches on SDVRP

	IP	**CP**				
No.of Vehicles	**No.of Variables**	**No.of Available Values for Each Variable**	**Complexity Function**	**No.of Variables**	**No.of Available Values for Each Variable**	**Complexity Function**
1	X : n^2	2	$2^{n^2} \cdot (\max[d_i]+1)^n$	X: n^2	2	$2^{n^2} \cdot (\max[d_i]+1)^n$
	A: n	$\max[d_i]+1$		A: n	$\max[d_i]+1$	
2	X: $2n^2$	2	$2^{2n^2} \cdot (\max[d_i]+1)^{2n}$	X: n^2	3	$3^{n^2} \cdot (\max[d_i]+1)^{2n}$
	A: 2n	$\max[d_i]+1$		A: 2n	$\max[d_i]+1$	
3	X: $3n^2$	2	$2^{3n^2} \cdot (\max[d_i]+1)^{3n}$	X: n^2	4	$4^{n^2} \cdot (\max[d_i]+1)^{3n}$
	A: 3n	$\max[d_i]+1$		A: 3n	$\max[d_i]+1$	
4	X: $4n^2$	2	$2^{4n^2} \cdot (\max[d_i]+1)^{4n}$	X: n^2	5	$5^{n^2} \cdot (\max[d_i]+1)^{4n}$
	A: 4n	$\max[d_i]+1$		A: 4n	$\max[d_i]+1$	

As seen from Table 5.4, complexity functions of the two approaches are similar when there is a single vehicle. As the number of vehicles increases, IP complexity increases much faster than CP complexity. Therefore, it is expected that IP approach provides much longer solution times than CP approach. However, in CP model for SDVRP (Model SDM) there exists both linear constraints and CP constraints. In this case, the model fails to provide any solution with two or more vehicles. Therefore, IP solutions are the only solutions that can be achieved.

The comparison between CP and IP approaches in terms of solution time and cost can be summarized in Figure 5.4.



	CP	IP
HVRP	OPTIMUM TO SUBPROBLEMS SOLUTION TIME = A	OPTIMUM TO SUBPROBLEMS SOLUTION TIME >> A
SDVRP	NO SOLUTION	OPTIMUM TO SUBPROBLEMS SOLUTION TIME >> A

Figure 5.4 Results of IP and CP approaches for HVRP and SDVRP

In summary, it can be said that CP approach is more suitable for HVRP since it provides solutions equal in cost but smaller in time compared to IP. On the other hand IP approach is more suitable for HVRP with split deliveries since no solution can be achieved by CP.

5.2 Test on Sample Instances of SDVRP

SDVRP has been first introduced by Dror and Trudeau (1989). The authors have showed the savings that can be achieved by allowing split deliveries (Archetti et al., 2006) and developed seven main test instances (Dror and Trudeau, 1994). These problems have also been solved by Archetti et. al. (2006).

In this dissertation, these seven test instances are considered in order to test the performance of Threshold Algorithm on SDVRP. The details of problem data are given in Table 5.5. Numbering schemes of the problems are similar to Archetti et. al. (2006). All vehicles are identical. For this reason vehicle fixed costs are not considered in the solution costs.

Table 5.5 Problem data of the seven test instances from Dror and Trudeau (1994)

Test Instance	Number of Nodes	Number of Vehicles	Capacity of Vehicles
1	50	6	160
2	75	10	140
3	100	8	200
4	150	12	200
5	199	16	200
6	120	7	200
7	100	10	200

The test instances listed in Table 5.5 are solved using the proposed approach in Section 5.1 (In the routing phase, integer programming is employed). The solutions belonging to Test Instance 1 are given in the following sections. Details of all other solutions can be found in the Appendix E.

5.2.1 Splitting into Subproblems

The threshold is set to be 26 at the first iteration. Then it is gradually increased by 2 units at each iteration. Model SC (Formulation 4.1, page 76) is solved to give three subsets at this threshold level. The iterations and the number of subproblems can be seen in Table 5.6. As seen from Table 5.6, when threshold is increased to 30 units, two subsets are formed. At this threshold level, one of the subproblems turn out to be very large (Number of nodes = 42). Therefore, the iterations stop at iteration 3. The algorithm goes on with threshold level 28 and the corresponding subsets.

Table 5.6 Threshold levels and the number of subproblems of Test Instance 1

Iteration	**Threshold**	**No. of subproblems**
1	26	3
2	28	3
3	30	2

5.2.2 Vehicle Assignment

Model SDVA (Formulation 5.2) is employed to solve Test Instance 1 of Dror and Trudeau (1994) to give three subproblems and the vehicles assigned to each one. The subproblems can be seen in Figure 5.5 (The red square denotes the depot node).

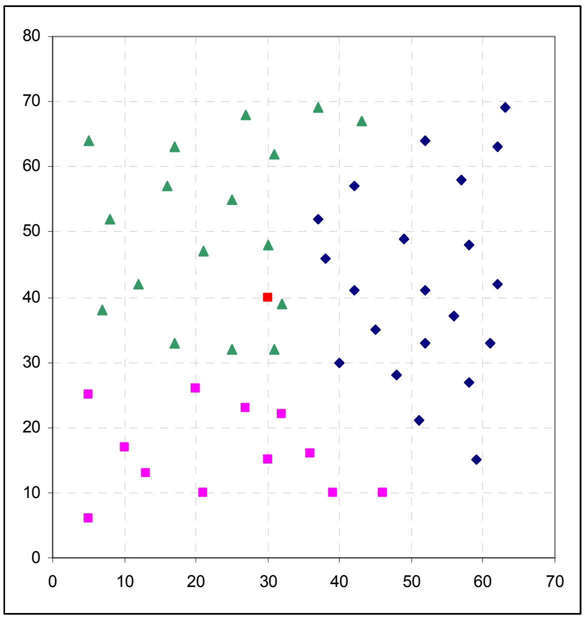


Figure 5.5 The two subproblems of test instance 1

5.2.3 Routing Phase

Each subproblem is solved by Model IPM (Formulation 5.4). The results at different threshold levels can be seen in Table 5.7. As expected, the best solution is achieved at threshold level 28 (higher threshold level). Therefore, this solution is the solution of Test Instance 1. The detailed solutions at threshold level 28 can be seen separately in Table 5.8.

Table 5.7 Solutions belonging to Test Instance 1 at different threshold levels

Iteration	**Threshold**	**No. of Subproblems**	**No. of Vehicles**	**Solution Cost**
1	26	3	6	571,86
2	28	3	5	536,13

Table 5.8 Solutions of subproblems belonging to Test Instance 1

Number of Nodes (n)	**Number of Vehicles (v)**	**Solution Cost**	**Computation Time (sec.)**
21	2	212,51	71
12	1	141,76	11
17	2	181,87	32

The subproblems are solved in parallel. Therefore, the computation time of the complete problem is the maximum of the computation times of subproblems. The computation time of Test Instance 1 is 71 seconds (max[71, 11, 32]). The routes of Test Instance 1 can be seen in Figure 5.6.

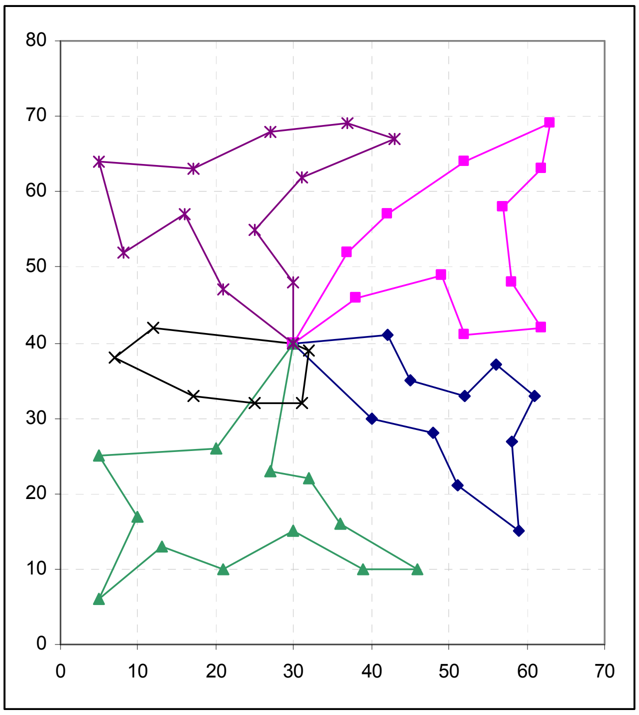


Figure 5.6 Routes of Test Instance 1

All test instances from Dror and Trudeau (1994) are solved. Detailed solutions can be seen in Appendix E.

5.3 Computational Results of Split Delivery Test Problems

All seven test instances from Dror and Trudeau (1994) are solved. These test problems are also handled by Archetti et. al. (2006). Archetti et. al. (2006) has proposed three alternative tabu search algorithms for SDVRP which are referred as “Split Tabu”, “Split Tabu-DT” and “Fast Split Tabu” respectively. The comparison of the results achieved in this dissertation with Archetti et. al. (2006) and Dror and Trudeau (1994) results are given in Table 5.9 and 5.10. The solutions in bold are the best known solutions published for these problems.

Table 5.9 Comparison of the Threshold Algorithm and the literature solutions on the solution cost

Problem	n	D & T (1994)	Archetti et. al. (2006)			Threshold Algorithm	Difference with the Best Known (%)
			SPLITABU	SPLITABU-DT	FAST-SPLITABU		
		z	$\overline{z}$	$\overline{z}$	$\overline{z}$	z	
1	50	587	**530**	534	534	536	1,13
2	75	895	852	**850**	**850**	884	3,92
3	100	902	846	**836**	**836**	877	4,76
4	150	1131	**1062**	1070	1088	1103	3,71
5	199	1376	1368	**1343**	1346	1392	3,53
6	120	108,4	108,5	**105,6**	**105,6**	106,2	0,55
7	100	952	**823**	825	825	825	0,24

Table 5.10 Comparison of the Threshold Algorithm and the literature solutions on the solution time

Problem	n	D & T (1994) (sec.)	Archetti et. al. (2006)		Threshold Algorithm (sec.)
			SPLITABU (sec.)	SPLITABU-DT (sec.)	
1	50	0	17	13	71
2	75	0	64	36	501
3	100	0	60	58	615
4	150	0	440	389	407
5	199	0	1900	386	563
6	120	1	40	38	229
7	100	0	86	49	59

As seen from Table 5.9, the algorithms developed by Archetti et. al. (2006) provide the best known solutions for these test problems. However, among the three algorithms developed by Archetti et. al. (2006), none of them dominates the others totally in terms of solution cost. When the results achieved by the Threshold Algorithm are considered, it is seen that none of the solution costs are better than the best known. However, they dominate all Dror and Trudeau (1994) solutions and some of the Archetti et. al. (2006) solutions.

On the other hand, when solution times are considered (Table 5.10), it can be stated that Threshold Algorithm offers solution times independent from the size of the problem. This is due to the clustering characteristic of the Threshold Algorithm. Since the main problem is split into subproblems, the size of the subproblems do not vary due to the size of the main problem. As the main problem enlarges, the number of subproblems increase but not their size. Therefore, the solution times do not vary much by the size of the main problem.

Also, it is seen from Table 5.10 that for test instances 6 and 7, the solution times are relatively small. This is because these problems are in natural small clusters. Hence it takes short time to solve these problems by the Threshold Algorithm. Solution costs achieved for these problems are also very close to the best known. Therefore, it can be said that Threshold Algorithm is especially beneficial for naturally clustered problems.

5.4 Modified Threshold Algorithm for HVRP with Split Deliveries

As stated earlier, to the best of our knowledge, none of the HVRP studies in the literature handle split delivery assumption and none of the SDVRP studies handle heterogeneous fleet assumption. Therefore, this dissertation is the first study which considers both heterogeneous fleet and split delivery assumptions together. The problem is referred as HVRP with split deliveries.

HVRP and SDVRP are handled on their own by the Threshold Algorithm in the previous sections of this dissertation. In this section HVRP with split deliveries is considered. HVRP with split deliveries brings the two assumptions (heterogeneous fleet of vehicles and allowing split deliveries) together in a single problem.

The Threshold Algorithm is modified according to the two assumptions of HVRP with split deliveries. In other words, the methods used to handle the heterogeneous fleet and the methods used to handle split deliveries are combined. When considering the heterogeneous fleet, the main problem is to work off the tradeoff between vehicle fixed costs and the number of vehicles. In order to solve this conflict an IFGP approach is proposed in the vehicle assignment phase of the Threshold Algorithm (Section 4.1.2, page 78). On the other hand, when split deliveries are allowed, an IP approach is proposed in the routing phase of the Threshold Algorithm (Section 5.1.3.2).

Therefore, when these two assumptions are brought together, the IFGP approach should be applied in the vehicle assignment phase (for the heterogeneous fleet assumption) and IP approach should be applied in the routing phase (for split delivery assumption) of the Threshold Algorithm. All other steps of the algorithm remain the same as in HVRP and SDVRP. The flow of the Threshold Algorithm for HVRP with split deliveries can be seen in Figure 5.7.

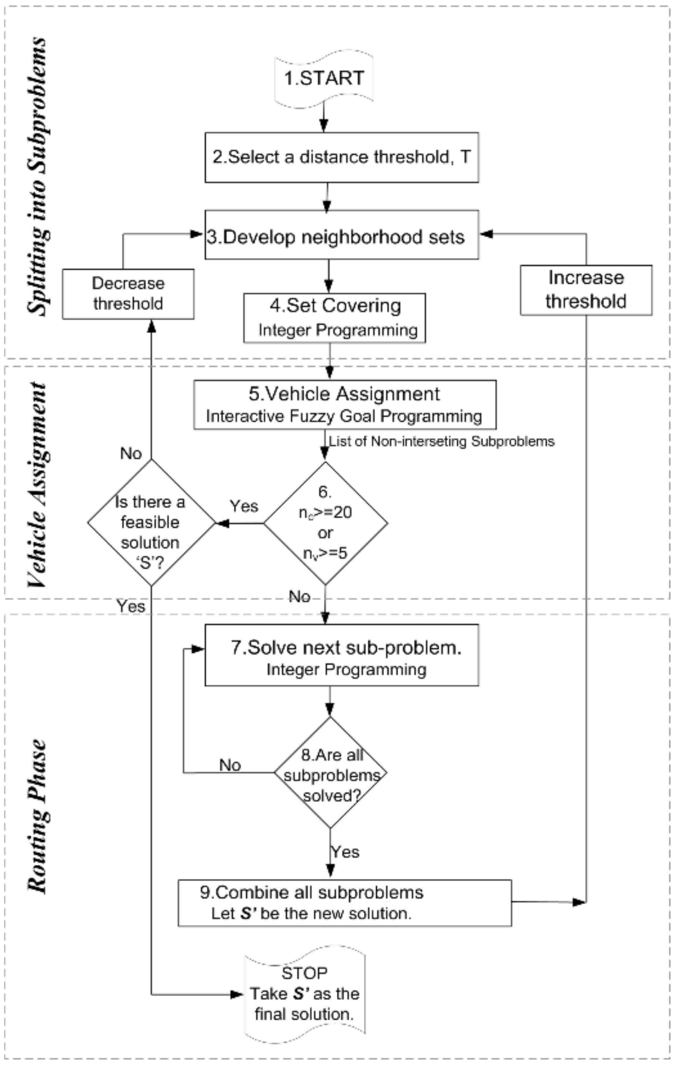

Splitting into Subproblems
1.START
2.Select a distance threshold, T
3.Develop neighborhood sets
Decrease threshold
Increase threshold
4.Set Covering Integer Programming
Vehicle Assignment
5.Vehicle Assignment Interactive Fuzzy Goal Programming
List of Non-interseting Subproblems
No
Is there a feasible solution 'S'?
Yes
6. n_c>=20 or n_v>=5
Yes
No
Routing Phase
7.Solve next sub-problem. Integer Programming
No
8.Are all subproblems solved?
Yes
9.Combine all subproblems Let S' be the new solution.
STOP Take S' as the final solution.

Figure 5.7 The flow of Threshold Algorithm for HVRP with split deliveries

5.5 Test on Sample Instances of HVRP with Split Deliveries

Since this is the first study which considers both heterogeneous fleet and split delivery assumptions in the literature, there exists no benchmark instances for this problem. Therefore, the benchmark instances belonging to HVRP (from Gendreau et. al., 1999) are solved under the split delivery assumption. It is known that allowing split deliveries is a relaxation of non-split delivery problem. Therefore, it is expected that solution costs under split delivery assumption would be at least as good as non-split delivery solution costs. The test problems (Test Instance 3 to Test Instance 20 from Gendreau et. al. 1999) are solved using the Threshold Algorithm proposed in Section 5.4. The results of HVRP with split deliveries are compared with the results of HVRP (achieved by the Threshold Algorithm in Chapter 4, page 96) in Table 5.10.

Table 5.10 Comparison of HVRP with split deliveries and HVRP

Problem No	Number of Nodes	Threshold Algorithm Non-Split Deliveries	Threshold Algorithm Split Deliveries	Deviation in Cost
3	20	983,1	970,53	-1,28%
4	20	6437,33	6421,88	-0,24%
5	20	1052,15	998,74	-5,08%
6	20	6516,47	6514,09	-0,04%
13	50	2440,78	2440,78	-0,00%
14	50	9138,25	9138,25	-0,00%
15	50	2620,54	2616,11	-0,17%
16	50	2746,3	2719,89	-0,96%
17	75	1783,33	1783,33	-0,00%
18	75	2394,16	2394,16	-0,00%
19	100	8734,35	8722,49	-0,14%
20	100	4139,80	4130,48	-0,23%

The third column in Table 5.10 lists the solutions achieved by the Threshold Algorithm for HVRP under non-split delivery assumption (from Chapter 4, page 96). The fourth column lists the solutions achieved under split delivery assumption. As seen from the table, by allowing split deliveries, better solution costs are achieved in instances 3, 4, 5, 6, 15, 16, 19 and 20. For instance 13, 14, 17 and 18, no improvement is achieved by split deliveries. That is all solution costs under split deliveries are either superior or equal to non-split delivery solution costs. This shows that allowing split deliveries in a distribution problem may lead to considerable decrease in distribution costs. The detailed solutions belonging to each test instance can be seen in Appendix F.

The solutions achieved for HVRP with split deliveries are also compared with the best known solutions in the literature. However, the best solutions in the literature assume non-split delivery strategy. As stated before, there are no benchmark instances and solutions considering both heterogeneous fleet and split deliveries together. Since allowing split deliveries in HVRP is a relaxation of HVRP, the solutions achieved in this dissertation are compared with the best known literature solutions for HVRP. The comparison is given in Table 5.11. The authors column lists the studies which have obtained the best known solution. T refers to Taillard (1999), GLMT refers to Gendreau et. al. (1999), and CT refers to Choi and Tcha (2007).

Table 5.11 Comparison of HVRP with split deliveries and best known literature solutions for HVRP

Problem No	**Number of Nodes**	**Best Known**	**Authors**	**Threshold Algorithm**	**Deviation in Cost**
3	20	961,03	T, GLMT, CT	970,53	0,98%
4	**20**	**6437,3**	**T, GLMT, CT**	**6421,88**	**-0,24%**
5	**20**	**1007,05**	**CT**	**998,74**	**-0,83%**
6	**20**	**6516,47**	**GLMT**	**6514,09**	**-0,04%**
13	50	2406,36	CT	2440,78	1,41%
14	50	9119,03	T, GLMT, CT	9138,25	0,21%
15	50	2586,37	T, GLMT, CT	2616,11	1,14%
16	**50**	**2720,43**	**CT**	**2719,89**	**-0,02%**
17	75	1744,83	CT	1783,33	2,16%
18	75	2371,49	CT	2394,16	0,95%
19	100	8661,81	T	8722,49	0,67%
20	100	4039,49	CT	4130,48	2,20%

As seen from Table 5.11, solution costs achieved in instances 4, 5, 6, and 16 dominate the solution costs of best known solutions. Solution costs achieved for the other test instances are within reasonable limits with the best known solutions. These test problems have not been considered under split delivery assumptions before. There is no information if the previous studies (Taillard 1999, Gendreau et. al. 1999, Choi and Tcha 2007) can handle split deliveries or not. Therefore, the solutions achieved for test instances 4, 5, 6 and 16 provide new benchmark values for the future studies in this area. In summary, according to the performance test results, it can be said that the Threshold Algorithm is also quite successful to handle and solve HVRP with split deliveries.

5.6 Fresh Goods Distribution of a Retail Store: Employing Split Delivery Strategy

The fresh goods distribution problem of the retail chain store (located in Izmir, Turkey) is handled employing the non-split delivery strategy in Chapter 4 of this dissertation. The problem is a fixed fleet HVRP. In this section, the same distribution problem is solved under split delivery assumption. In other words, the split delivery distribution strategy is employed for the retail chain store fresh goods distribution.

Splitting the problem into subproblems and vehicle assignment of the split delivery case is similar to the non-split delivery case. Since split deliveries are allowed in the routes of vehicles, only the routing phase differs. Therefore, the subproblems and the vehicles assigned to each subproblem are just as same as HVRP (under non-split delivery assumption, from Chapter 4). The subproblems can be seen in Figure 5.8.

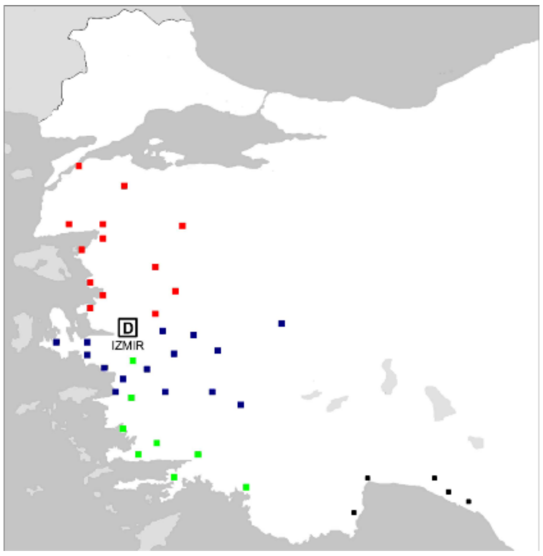


Figure 5.8 Subproblems of the retail chain store distribution problem

5.6.1 Routings Under Split Delivery Strategy

All subproblems are solved by Model IPM (Formulation 5.4). The results of non-split delivery and split delivery strategies together with the current performance of the retail chain store can be seen in Table 5.12. (Mizrak Ozfirat, Ozkarahan, 2007).

Table 5.12 Comparison of different distribution strategies and the current performance of the firm

	No. of Vehicles Allocated	**Fixed Cost (YTL)**	**Traveling Cost (YTL)**	**Total Cost (YTL)**	**Solution Time (sec.)**	**Improvement (%)**
Current Performance of the firm	5	5362	3908	9270	—	—
Non-split deliveries	**4**	4311	3792	8103	36	12,59
Split deliveries	**4**	**4311**	**3705**	**8016**	**657**	**13,53**

As seen from Table 5.12, both strategies decrease the current distribution costs of the firm in a considerable way. However, split delivery strategy leads to 13,53% improvement in the current distribution performance and dominates non-split delivery strategy in terms of solution cost. This is expected in the sense that splitting deliveries is relaxation of the problem.

The improvement is achieved in the traveling cost. In other words, routes have changed so that traveling costs have decreased. The routings can be seen in Figure 5.9. However, when solution times are considered, it is seen that solution under split delivery strategy is obtained in a much longer time than non-split delivery strategy solution. This is mainly due to the approach employed. CP is employed for the non-split delivery strategy whereas IP is employed for the split delivery strategy. Therefore, it can be said that CP is able to achieve faster solutions compared to IP.

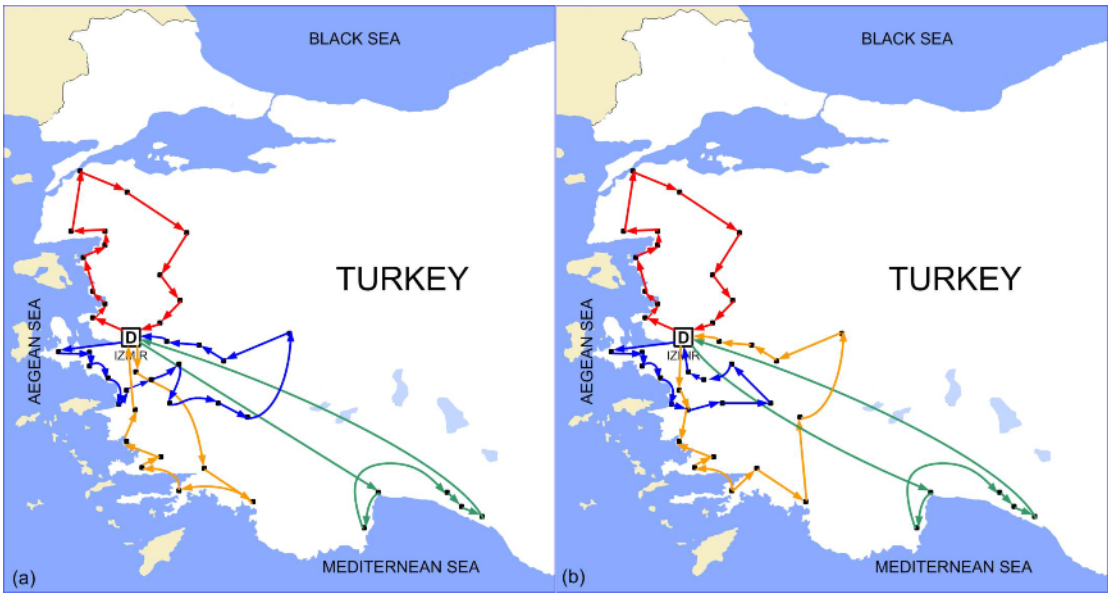


Figure 5.9 Routes of retail chain store according to (a) non-split deliveries (b) split deliveries

Looking at Figure 5.9, it can be seen that there is only one demand node which is visited by two vehicles. That is, demand of only one node is split between two vehicles. However, the overall solution cost turns out to have improved. In summary, by allowing split deliveries, even a slight change in the routes may lead to considerable improvement in the solution cost. Therefore, the solutions achieved in this dissertation are presented to the retail chain store and split delivery strategy is advised to the firm.

CHAPTER SIX
THRESHOLD ALGORITHM FOR VRP WITH TIME WINDOWS

Vehicle routing problems are important and well-known combinatorial optimization problems occurring in many transport logistics and distribution systems of considerable economic significance. The vehicle routing problem with time windows (VRPTW) has recently received a lot of attention in the literature. Firstly, because VRPTW is still one of the most difficult problems in combinatorial optimization and consequently presents a great challenge. Secondly, because of the wide applicability of time window constraints in real-world cases. Mainly, due to these two reasons, the objective of this chapter is develop an effective procedure for VRPTW which can be both employed in practical real life applications and theoretical problems in the literature.

In the VRPTW, the objective is to find a set of minimum-cost vehicle routes, which start at a central depot, service a set of customers with known demands, and return to the depot. Each customer must be serviced once by a vehicle, and the total demands of the customers serviced by the vehicle must not exceed the capacity of the vehicle. Moreover, each customer must be serviced within a specified time window. If a vehicle arrives at a customer earlier than the lower bound of the customer's time window, the vehicle must wait until the service is possible. In some applications, service after the upper bound of the time window is also allowed, but a penalty is set for the service. The depot has also a time window, and all the vehicles must return by the closing time of the depot. The objective is to minimize the number of tours or routes, and then for the same number of routes, to minimize the total traveled distance. VRPTW has a wide range of applications such as Chinese Postman Problem, bank deliveries, school bus routing and so on.

Within this chapter, the Threshold Algorithm is modified according to time window assumptions and its performance on VRPTW is tested. In the first section of the chapter, the modifications made on the Threshold Algorithm are defined. However, clustering phase of the algorithm turned out to be inapplicable when time windows are incorporated into the problem. Therefore, Threshold Algorithm can only be employed when the problem is already in clusters. In other words, a large scale VRPTW cannot be split into clusters and hence, it is not solvable by the Threshold Algorithm.

Since, large scale VRPTWs cannot be handled by the Threshold Algorithm, a set covering based algorithm is developed for these type of problems. This algorithm is defined in Section 6.1.2. In Figure 6.1, the classification made on VRPTW and the proposed algorithms in this chapter can be seen.

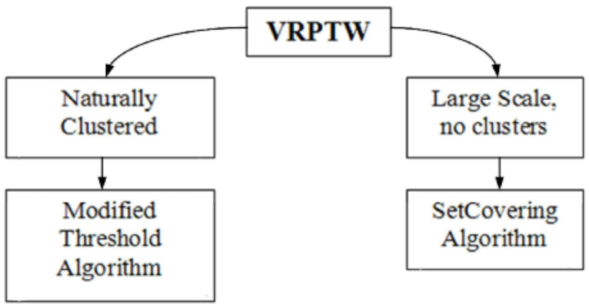


Figure 6.1 Classification on VRPTW and the corresponding algorithms proposed in this dissertation

The performance of the proposed algorithms are tested on the well known Solomon benchmark instances from the literature (Cordeau et. al., 2002). Computational results of performance tests are given in Section 6.2. Finally, concluding remarks on VRPTW are given in Section 6.3.

6.1 Proposed Algorithm for VRPTW

Due to the special characteristics of time windows, VRPTW is considered under two cases as clustered problems and large scale problems. For clustered problems, the Threshold Algorithm is modified according to time window assumptions. On the other hand, for large scale VRPTW, a novel SetCovering Algorithm is developed. Both algorithms are explained in detail in this section.

6.1.1 Threshold Algorithm Modifications

The Threshold Algorithm, which is used to handle HVRP and SDVRP, is modified according to time window assumptions. However, there exist some problems in the clustering phase of the algorithm due to the special characteristics of time window assumptions.

In VRPTW, each node has a certain time window. When dividing the main problem into subproblems, time window requirements should be consistent with each other. Otherwise subproblem would turn out to be infeasible. In the clustering phase of the algorithm, however, there is no way to consider time windows of nodes. This is due to the decision variables defined in the clustering phase, which are (Refer to Table 4.2 from Chapter 4, page 80):

$$y_{ij} : \begin{cases} 1 & \textit{if node i is assigned to subproblem j.} \\ 0 & \textit{otherwise} \end{cases}$$

$$w_{vj} : \begin{cases} 1 & \textit{if vehicle v is assigned to subproblem j.} \\ 0 & \textit{otherwise} \end{cases}$$

There are no time and distance dimensions in the clustering phase. In order to handle time windows, decision variables, which define routes, are necessary such as:

$$X_{ijv} : \begin{cases} 1 & \textit{if vehicle v travels from node i to node j.} \\ 0 & \textit{otherwise} \end{cases}$$

However, this decision variable is the subject of the routing phase. Therefore, it is not possible to handle VRPTW with cluster first route second algorithms.

The clustering phase is not applicable. However, as long as the problem size is convenient, VRPTW can be solved by constraint programming (CP) approach in the routing phase of the proposed algorithm. The size of the problems cannot be decreased by splitting up the problem. But, VRPTW has also some advantages to reduce the search space and hence decrease the problem size. Since, all nodes require to be visited within a certain time window, some of the arcs on the graph are eliminated due to inconsistent time windows. In order to define these inconsistencies, some of the notation belonging to VRPTW should be given:

$Earliest_i$: *Earliest time a vehicle can arrive to node i.*

$Latest_i$: *Latest time a vehicle can arrive to node i.*

$Service_i$: *Service time at node i.*

The inconsistencies according to time window requirements can be defined as follows:

i. If the latest time of node ***j*** is smaller than the earliest time of node ***i***, then no vehicle is allowed to travel from node ***i*** to node ***j***.

Example:

Node	**Earliest Time**	**Latest Time**
i	110	150
j	10	90

In this case no vehicle is allowed to travel from node ***i*** to node ***j*** since $Latest_j < Earliest_i$.

ii. If latest time of node ***j*** is smaller than the sum of leaving time of node ***i*** and traveling time from node ***i*** to node ***j***, then no vehicle is allowed to travel from node ***i*** to node ***j***.

Example:

Node	**Earliest Time**	**Latest Time**	**Service Time**	**Traveling Time**
				(i to j)
i	110	150	10	20
j	100	130	10	20

The earliest possible time to leave node ***i*** = $Earliest_i + Service_i = 120$

The earliest possible arrival time to node ***j*** from node ***i*** = 120+ Traveling Time=140

Since the earliest possible arrival time to ***j*** from ***i*** is greater than $Latest_i$ (***140>130***), no vehicle is allowed to travel from node ***i*** to node ***j***.

Since the second case covers the first one, only the second case is set as a rule. Considering this rule, a matrix ***M*** is built where,

$$M_{ij} : \begin{cases} 1 & if\ Earliest_i + Service_i + Traveling_{ij} \leq Latest_j \\ 0 & otherwise \end{cases}$$

Using matrix ***M***, the search space of variables can be reduced. Considering time window assumptions and the inconsistency rule, a constraint programming (CP) model, called Time Window Model (Model TWM), is built. The notation used in the model is given in Table 6.1 and the model is given in Formulation 6.1. As stated previously, the clustering phase of the Threshold Algorithm is inapplicable to VRPTW. Routing phase is applied directly. Model TWM is the CP model of the routing phase of Threshold Algorithm, which is modified according to time window assumptions (Mizrak Ozfirat and Ozkarahan, 2008a).

Table 6.1 Notation used in model TWM

Parameters :
n_c : *Number of nodes to be served.*
n_v : *Number of vehicles.*
$Demand_j$:*Demand of node* $j, j : 0..n_c$.
$Routecap_j$: *Total demand of route* j.
Cap_v : *Capacity of vehicle* $v, v : 1..n_v$.
D_{ij} : *Distance from node i to node* $j, i : 0..n_c, j : 0..n_c$.
$Latest_i$: *Latest time a vehicle can arrive to node* $i, i : 0..n_c$.
$Earliest_i$: *Earliest time a vehicle can arrive to node* $i, i : 0..n_c$.
$Service_i$: *Service time at node* $i, i : 0..n_c$.
$Traveling_{ij}$: *Traveling time from node i to node* $j, i : 0..n_c, j : 0..n_c$.
$M_{ij} : \begin{cases} 1 & \text{if } Earliest_i + Service_i + Traveling_{ij} \leq Latest_j \\ 0 & \text{otherwise} \end{cases} ,i : 0..n_c, j : 0..n_c.$
$Ready$: *Latest time a vehicle should complete its tour.*
Sets :
$Stores = \{1,2,....,n_c\}$
$Total = Stores \cup \{0\}$; "0" *denotes depot node.*
$Vehicles = \{1,2,.....,n_v\}$
Decision Variables :
$X_{ij} : \begin{cases} \text{Number of the vehicle that travels from node } i \text{ to node } j. \\ 0 \quad \text{if no vehicle travels from node } i \text{ to node } j. \end{cases} ,i : 0..n_c, j : 0..n_c.$
T_i : *Arrival time to node* $i,\ i : 0..n_c$.
W_i : *Waiting time at node* $i = \begin{cases} (Earliest_i - T_i) & \text{if } T_i \prec Earliest_i \\ 0 & \text{otherwise} \end{cases} ,i : 0..n_c.$

$Model\ TWM:$

$Solve$

$$\sum_{j \in Total} (X_{ij} > 0) = 1 \qquad \forall i \in Stores \qquad (6.1a)$$

$$\sum_{i \in Total} (X_{ij} > 0) = 1 \qquad \forall j \in Stores \qquad (6.1b)$$

$$\sum_{i \in Total} (X_{i0} = v) = 1 \qquad \forall v \in Vehicles \qquad (6.1c)$$

$$\sum_{j \in Total} (X_{0j} = v) = 1 \qquad \forall v \in Vehicles \qquad (6.1d)$$

$$\sum_{j \in Total} X_{ij} = \sum_{j \in Total} X_{ji} \qquad \forall i \in Total \qquad (6.1e)$$

$$\sum_{i \in Total} \sum_{j \in Total, j \neq i} (X_{ij} = v) * Demand_j \leq Cap_v \qquad \forall v \in Vehicles \qquad (6.1f)$$

$$T_i + W_i \leq Latest_i \qquad \forall i \in Stores \qquad (6.1g)$$

$$T_i + W_i \geq Earliest_i \qquad \forall i \in Stores \qquad (6.1h)$$

$$\sum_{v \in Vehicles} \sum_{i \in Total} \left(X_{ij} = v\right) * \left(T_i + W_i + Service_i + Traveling_{ij}\right) \leq T_j \qquad \forall j \in Stores \qquad (6.1i$$

$$\sum_{j \in Stores} \left(X_{j0} = v\right) * \left(T_j + W_j + Service_j + Traveling_{j0}\right) \leq Ready \qquad \forall v \in Vehicles \qquad (6.1j$$

$$T_0 = 0 \qquad (6.1k)$$

$$W_0 = 0 \qquad (6.1l)$$

$$\left(X_{ij} > 0\right) \leq M_{ij} \qquad \forall i, j \in Total \qquad (6.1m)$$

(6.1)

Constraints (***6.1a***) and (***6.1b***) assure that each customer is visited exactly once. In addition, each vehicle should visit the depot once. This is included in the model with constraints (***6.1c***) and (***6.1d***). Constraint set (***6.1e***) provides that a vehicle visiting a customer should leave that customer. It is stated by constraint set (***6.1f***) that capacity of vehicles should not be exceeded. Constraints (***6.1g***) and (***6.1h***) assure that each node is visited within its specified time window. Constraints (***6.1i***) state that arrival time to a node must be greater than or equal to the sum of arrival time to the previous node, waiting time and service time there and the traveling time in between. Similar to these constraints, (***6.1j***) states that arrival time to the depot node must be less than or equal to the maximum tour completion time (***Ready***). Initialization of ***T*** and ***W*** variables are included in the model with (***6.1k***) and (***6.1l***). Finally, constraints (***6.1m***) assure X_{ij} to take the value of "0" if it is not possible to move from ***i*** to ***j*** due to their time windows (due to inconsistent time windows).

In Formulation 6.1, the objective function and the subtour elimination constraints are missing. Subtour elimination constraints are handled with the Subtour Elimination Algorithm (SEA) (previously employed for HVRP in Chapter 4 of this dissertation, page 89). The flow of the SEA can be seen in the following.

OPL Script SEA:

```
1. Start;
2. MinimumDistance:=MaximumInteger;
3. Let S be the next solution of model TWM;
4. If S≠∅ then 5, else 9;
5. Distance:=Total Distance Traveled in S;
6. If S is a full solution (does not contain subtours), then 7, else 8
7. If  Distance<MinimumDistance, then MinimumDistance:=Distance;
8. Go to 3;
9. Stop;
```

SEA handles the first possible solution of the Model TWM. It controls whether the solution consists of a subtour. If it is a full solution (does not contain any subtours), the value of total distance traveled is recorded as ***MinimumDistance.*** Then the algorithm takes the next possible solution (of Model TWM) and makes subtour controls. If it is not a subtour and the total distance traveled belonging to this solution is less than ***MinimumDistance***, it is recorded as the new ***MinimumDistance***. The procedure goes on in the same way until all possible solutions are handled. Once enumeration is completed, the final ***MinimumDistance*** value turns out to be the optimum solution of the problem. Shortly, it can be said that SEA makes complete enumeration among the feasible solutions of Model TWM and finds the minimum distance routes which do not contain any subtours. The OPL code of Model TWM is given in Appendix G (OPL code of SEA can be found in Appndix A).

6.1.2 Set Covering Based Algorithm

As stated before, Model TWM in the proposed algorithm can only be employed as long as the size of the problem permits. When the problem size increases, it becomes unsolvable with the Threshold Algorithm. Therefore, for larger scale problems, a set covering based algorithm is developed. The algorithm depends on covering all nodes only by using allowable movements. An allowable movement from one node to another is defined by the time window requirements of the two nodes. Movement from ***i*** to ***j*** is allowed only if

$$Earliest_i + Service_i + Traveling_{ij} \leq Latest_j$$

The matrix ***M***, which defines the allowable movements according to time windows, is built (as in Section 6.1.1).

$$M_{ij} : \begin{cases} 1 & if\ Earliest_i + Service_i + Traveling_{ij} \leq Latest_j \\ 0 & otherwise \end{cases}$$

A binary IP model, called the Model SetCovering, is built, which solves a set covering problem on matrix ***M***. The notation used in the model is given in Table 6.2. Model SetCovering is given in Formulation 6.2.

Table 6.2 Notation used in Model SetCovering

$Parameters:$
n_c : *Number of nodes to be served.*
n_v : *Number of vehicles.*
D_{ij} : *Distance from node i to node j, i : 0..n_c, j : 0..n_c.*
$Latest_i$: *Latest time a vehicle can arrive to node i, i : 0..n_c.*
$Earliest_i$: *Earliest time a vehicle can arrive to node i, i : 0..n_c.*
$Service_i$: *Service time at node i, i : 0..n_c.*
$Traveling_{ij}$: *Traveling time from node i to node j, i : 0..n_c, j : 0..n_c.*
$M_{ij} : \begin{cases} 1 & if\ Earliest_i + Service_i + Traveling_{ij} \leq Latest_j \\ 0 & otherwise \end{cases} ,i : 0..n_c, j : 0..n_c.$
$Sets:$
$Stores = \{1,2,....,n_c\}$
$Total = Stores \cup \{0\}$; "0" *denotes depot node.*
$Decision\ Variables:$
$Y_{ij} : \begin{cases} 1 & if\ arc\ i\ to\ j\ is\ selected \\ 0 & otherwise \end{cases} ,i : 0..n_c, j : 0..n_c.$

$$
\begin{array}{llll}
\textit{Model SetCovering}: & & & \\
\textit{minimize} \displaystyle\sum_{i\in Total}\sum_{j\in Total} Y_{ij} * D_{ij} & & \textit{(6.2a)} & \\
\textit{subject to} & & & \\
\displaystyle\sum_{i\in Total} Y_{ij} = 1 & \forall j \in Stores & \textit{(6.2b)} & \\
\displaystyle\sum_{j\in Total} Y_{ij} = 1 & \forall i \in Stores & \textit{(6.2c)} & \textbf{(6.2)} \\
\displaystyle\sum_{j\in Total} Y_{0j} = \sum_{i\in Total} Y_{i0} & & \textit{(6.2d)} & \\
\displaystyle\sum_{j\in Total} Y_{0j} \le n_v & & \textit{(6.2e)} & \\
Y_{ij} \le M_{ij} & \forall i, j \in Total & \textit{(6.2f)} & \\
Y_{ij} \in \{0,1\} & & \textit{(6.2g)} &
\end{array}
$$

The model tries to minimize total distance traveled (***6.2a***) while covering all nodes exactly once (***6.2b***, ***6.2c***). In addition, constraint (***6.2d***) state that the number of arcs (vehicles) leaving the depot node should be equal to the number of incoming arcs.

According to this, (***6.2e***) assures the number of arcs outgoing from the depot node should not exceed the number of available vehicles. Finally, constraints (***6.2f***) provide only to select the allowable movements from matrix ***M***.

Within Model SetCovering, subtours, capacity requirements, and maximum tour completion requirement are not considered. These are not considered within the model in order to decrease problem size and to speed up the procedure. All of these concepts are handled by an OPL script algorithm given below.

OPL Script Algorithm:

```
1. Start;
2. Solve Model SetCovering, let S be the solution;
3. If S does not contain any subtours then go to 7;
4. Identify subtours;
5. Add corresponding subtour elimination constraints to Model SetCovering;
6. Go to 2;
7. Identify overcapacities in S;
8. Add corresponding overcapacity elimination constraints to Model SetCovering;
9. Identify overtimes in S;
10. Add corresponding overtime elimination constraints to Model SetCovering;
11. Identify time window violations in S;
12. Add corresponding time window violation elimination constraints to Model
    SetCovering;
13. If there are no overcapacity, overtime and time window violations go to 15;
14. Go to 2;
15. STOP, S is the optimum solution;
```

The script algorithm solves the Model SetCovering. The solution is identified whether it contains any subtours or not. If subtours exist then corresponding subtour constraints are included in Formulation 6.2 and the model is resolved. The model is solved iteratively in this manner until a solution without any subtours is achieved (Steps 2 to 6). When a full solution is achieved, it is controlled in terms of overcapacities, overtimes and time window violations. If these are identified, the corresponding constraints are added back into Formulation 6.2 and the model is resolved.

To integrate the script algorithm and the SetCovering model, additional constraints, which belong to subtours, overcapacities, overtimes and time windows, are included into Formulation 6.2. Necessary notation is given in Table 6.3.

Table 6.3 Additional notation for SetCovering model

Parameters :

Noofsubtours: No of subtours identified by the OPL Script $A\lg orithm$.

Noofovercapacity: No of overcapacities identified by the OPL Script $A\lg orithm$.

Noofovertime: No of overtime identified by the OPL Script $A\lg orithm$.

Nooftwviolation: No of time window violations identified by the OPL Script $A\lg orithm$.

$subtours_{ti}$: *The* i^{th} *node which is included in subtour* $t, t : 1..Noofsubtours$.

$exceedingdemand_{ti}$: *The* i^{th} *node which is included in overcapacity* $t, t : 1..Noofovercapacity$.

$exceedingtime_{ti}$: *The* i^{th} *node which is included in overtime* $t, t : 1..Noofovertime$.

$outoftw_{ti}$: *The* i^{th} *node which is included in time window violation* $t, t : 1..Nooftwviolation$.

rhs_t*:Maximum allowable value to break subtour t, t:*1*..Noofsubtours.*

Capacity:Capacity of vehicles .

$rhs2_t$*:Maximum allowable value to avoid overtime t, t:*1*..Noofovertime.*

$rhs3_t$*:Maximum allowable value to avoid time window violation, t:*1*..Nooftwviolation.*

Sets :

$Subtourrange = \{1,2,...,No.ofsubtours\}$

$Overcaprange = \{1,2,...,No.ofovercapacity\}$

$Overtimerange = \{1,2,...,No.ofovertime\}$

$Outrange = \{1,2,...,No.oftwviolation\}$

Additional constraints to SetCovering model to work together with OPL Script algorithm:

$$\sum_{i \in \{1,..,n_c-1\}} Y_{subtours_{t\,i} subtours_{t\,i+1}} \leq rhs_t \quad \forall t \in Subtourrange \quad (6.2h)$$

$$\sum_{i \in \{1,..,n_c-1\}} Y_{subtours_{t\,i+1} subtours_{t\,i}} \leq rhs_t \quad \forall t \in Subtourrange \quad (6.2i)$$

$$\sum_{i \in \{0,..,n_c-1\}} \left(Y_{exceedingdemand_{t\,i} exceedingdemand_{t\,i+1}}\right) * Demand_{exceedingdemand_{ti}} \leq Capacity \quad \forall t \in Overcaprange \quad (6.2j)$$

$$\sum_{i \in \{0,..,n_c-1\}} \left(Y_{exceedingdemand_{t+1\,i} exceedingdemand_{t\,i}}\right) * Demand_{exceedingdemand_{ti}} \leq Capacity \quad \forall t \in Overcaprange \quad (6.2k)$$

$$\sum_{i \in \{0,..,n_c-1\}} Y_{exceedingtime_{t\,i} exceedingtime_{t\,i+1}} \leq rhs2_t \quad \forall t \in Overtimerange \quad (6.2l)$$

$$\sum_{i \in \{0,..,n_c-1\}} Y_{exceedingtime_{t\,i+1} exceedingtime_{t\,i}} \leq rhs2_t \quad \forall t \in Overtimerange \quad (6.2m)$$

$$\sum_{i \in \{0,..,n_c-1\}} Y_{outoftw_{t\,i} outoftw_{t\,i+1}} \leq rhs3_t \quad \forall t \in Outrange \quad (6.2n)$$

$$\sum_{i \in \{0,..,n_c-1\}} Y_{outoftw_{t\,i+1} outoftw_{t\,i}} \leq rhs3_t \quad \forall t \in Outrange \quad (6.2p)$$

By this procedure, all the subtour, capacity, overtime and time window constraints are excluded from the model at the very first iteration. Hence the problem size is reduced in a considerable way. Then at every iteration, the solution is checked for all these violations. If there exist any of them, then the belonging constraints are added back into Model SetCovering and the model is resolved. With this algorithm, only the constraints (subtour, capacity, time window and overtime), which affect the global optimum, are considered. All other constraints are excluded and the problem is reduced to a solvable size.

(***6.2h***) and (***6.2i***) are subtour elimination constraints. (***6.2j***) and (***6.2k***) are overcapacity constraints. Overtime constraints are included by (***6.2l***) and (***6.2m***). Finally, constraints (***6.2n***) and (***6.2p***) belong to time window violations. The flow of algorithm can be seen in Figure 6.2. The OPL Script code is given in Appendix G.

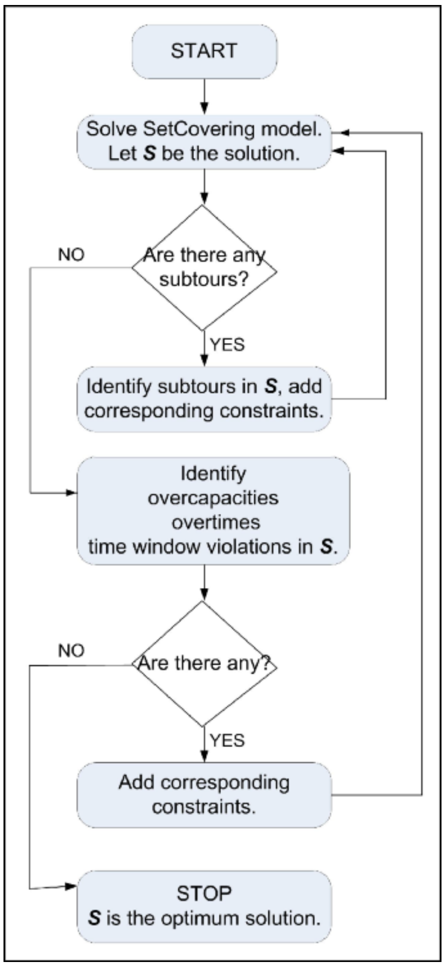

START
Solve SetCovering model. Let ***S*** be the solution.
Are there any subtours?
NO
YES
Identify subtours in ***S***, add corresponding constraints.
Identify overcapacities overtimes time window violations in ***S***.
Are there any?
NO
YES
Add corresponding constraints.
STOP ***S*** is the optimum solution.

Figure 6.2 Flow of SetCovering Algorithm

SetCovering Algorithm provides the global optimum. That is, if a problem is solvable by the SetCovering Algorithm, then the solution achieved to that problem is the global optimum. This is shown in the following section (Mizrak Ozfirat and Ozkarahan, 2008b).

6.1.2.1 Demonstrating that SetCovering Algorithm Provides Global Optimum

In a VRPTW mathematical model, all of the subtour, overcapacity, overtime and time window violation constraints should exist in order to achieve a feasible solution. However, at the very first step of the SetCovering Algorithm, all of these constraints are excluded from the SetCovering Model (Formulation 6.2, page 139). Then, every solution achieved by the SetCovering Model is controlled by the script algorithm whether it contains any of these violations. If there are any, then the solution achieved is not feasible. And hence, these are included back into the model as constraints. All the constraints included are valid cuts of the model. The model is solved iteratively in this manner until a feasible solution is achieved. This feasible solution is also the global optimum since the total traveling cost is minimized as the objective.

- Firstly, consider a VRPTW with 2 demand nodes where "0" denotes the depot node.

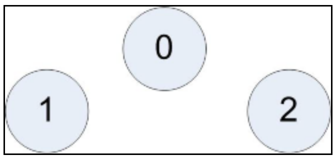


Figure 6.3 Two node VRPTW example problem

There can be only two subtours in this problem and both contain only a single node. These are 1-1 and 2-2 subtours as shown in Figure 6.4a and 6.4b.

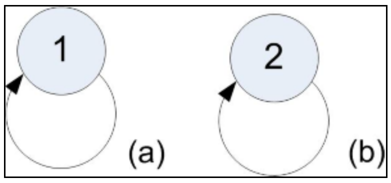


Figure 6.4 Possible subtours of a two node VRPTW

The constraints which break these two subtours should exist in the mathematical model of a VRPTW.

- Next, consider a VRPTW with 3 demand nodes where "0" denotes the depot node.

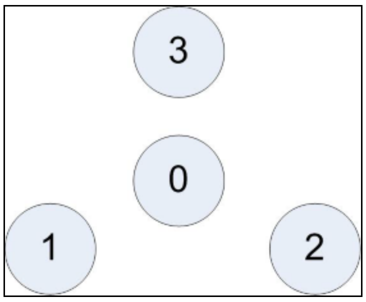


Figure 6.5 Three node VRPTW example problem

In this problem, there can be three subtours with a single node and three subtours with two nodes. These are:

Subtours of 1 node	Subtours of 2 nodes
1-1	1- 2-1
2-2	1- 3-1
3-3	2- 3-2

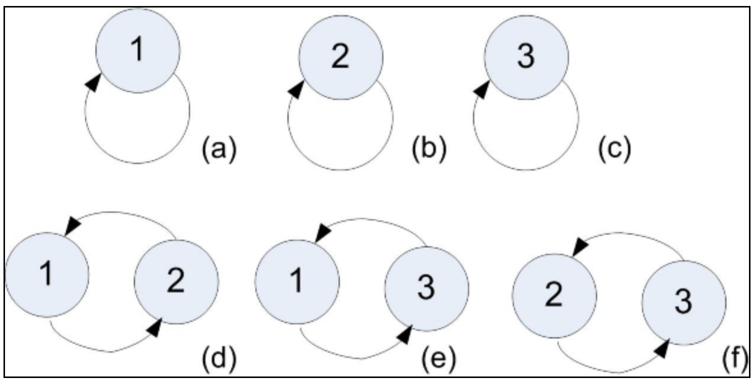


Figure 6.6 Possible subtours of a three node VRPTW

Representation of the subtours can be seen in Figure 6.6. Similar to the two-nodes problems, the constraints which break these subtours should exist in the mathematical model of a VRPTW.

- These examples can be generalized to a problem of ***n*** nodes. In Table 6.4, the number of subtours according to the number of nodes in the problem can be seen.

Table 6.4 Number of subtours according to the number of demand nodes

Number of Nodes (i)	**Subtour of 1 node**	**Subtour of 2 nodes**	**Subtour of 3 nodes**	**Subtour of 4 nodes**	**Subtour of 5 nodes**	**...**	**Subtour of (n-1) nodes**
2	$\binom{2}{1}$	-	-	-	-	-	-
3	$\binom{3}{1}$	$\binom{3}{2}$	-	-	-	-	-
4	$\binom{4}{1}$	$\binom{4}{2}$	$\binom{4}{3}$	-	-	-	-
5	$\binom{5}{1}$	$\binom{5}{2}$	$\binom{5}{3}$	$\binom{5}{4}$	-	-	-
....						-	-
n	$\binom{n}{1}$	$\binom{n}{2}$	$\binom{n}{3}$	$\binom{n}{4}$	$\binom{n}{5}$	...	$\binom{n}{n-1}$

As seen from Table 6.4, number of subtour constraints increase quadratically as number of nodes increase. When there are ***n*** demand nodes in the problem, including all subtour constraints makes the problem np-hard and hence, it may be insolvable. Therefore, in the SetCovering Algorithm proposed in this dissertation, the aim is to decrease problem size by removing the constraints, which in fact do not affect the optimum solution.

In the traditional VRPTW mathematical models, together with all subtour constraints, a feasible region is constituted for the problem. Then the model searches this region to find the optimum solution. A representative figure of the feasible region is given in Figure 6.7a. Assume that Lines 1 to 5, which form the feasible region (shaded region in Figure6.7a) belong to subtour constraints. The optimum solution is within this region. SetCovering Algorithm is able to find the optimum solution within this region faster.

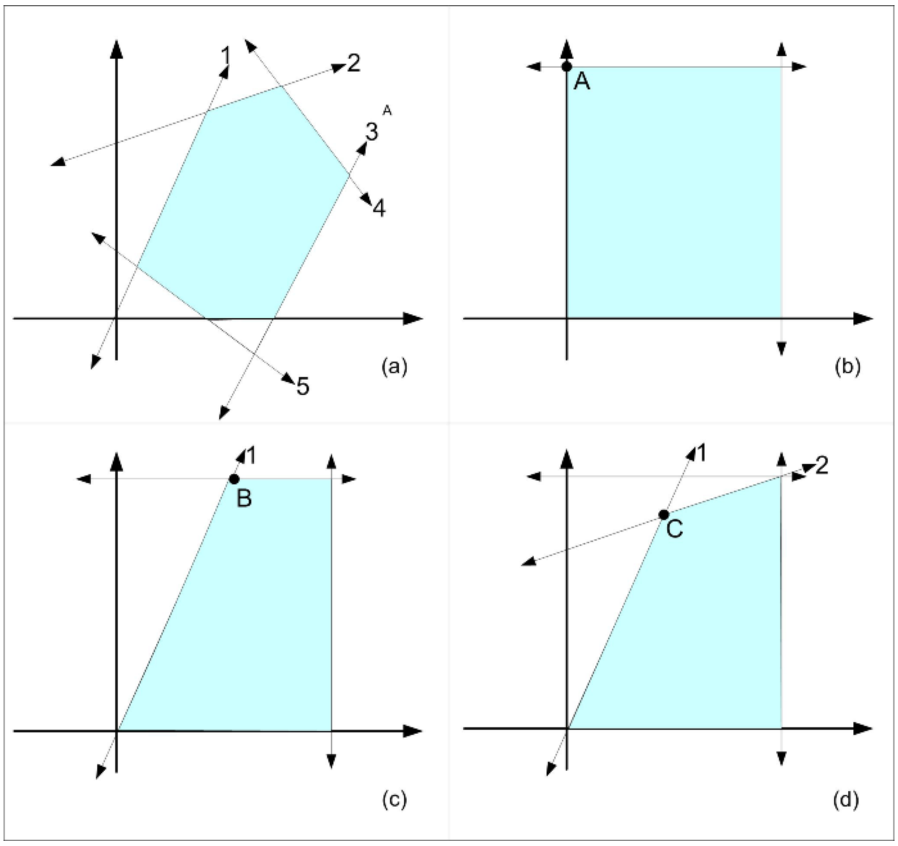


Figure 6.7 Demonstration of how SetCovering Algorithm works

The flow of the SetCovering algorithm is explained by a demonstrative example:

i. In the SetCovering Algorithm, by excluding all the subtour constraints, the feasible region is enlarged (Figure 6.7b). Model SetCovering (Formulation 6.2, page 139) is solved. Let the optimum solution be at Point A. However, this point is not feasible for the original problem (Figure 6.7a) .

ii. The script algorithm identifies the subtours, which causes infeasibility and adds these constraints back into Model SetCovering (Line 1 in Figure 6.7c). The feasible region becomes as in Figure 6.7c. Model SetCovering is resolved. Let the new optimum solution be Point B, which is again infeasible.

iii. The script algorithm adds again the necessary subtour constraints back into Model SetCovering (Line 2 in Figure 6.7d). The new feasible region becomes as in Figure 6.7d. Model SetCovering is solved once more. Let the optimum solution be Point C. This solution is also feasible according to the original feasible region (Figure 6.7a). Therefore, it is the global optimum solution of the problem.

As seen from the example, by use of SetCovering Algorithm, on the way to the optimum solution, some of the subtour constraints are not considered at all (Lines 3, 4 and 5 from Figure 6.7a) since they do nor affect the optimum solution. In other words, the advantage of SetCovering Algorithm is to decrease problem size and lead to faster solutions. Overcapacity, overtime and time window constraints are treated just as similar to subtour constraints. That is, the overcapacity, overtime and time window constraints, which do not affect the optimum solution, are not considered at all. In conclusion, it can be said that, when a solution is achieved by the SetCovering Algorithm, it is the global optimum solution.

6.2 Performance Tests on Benchmark Problems

The routing part of the Threshold algorithm (Section 6.1.1) can be employed when the problem is already in clusters. In other words, a time windowed VRP cannot be split into clusters. But if the problem is naturally clustered then routing procedure can be applied. For other problems, the SetCovering Algorithm is employed (Section 6.1.2).

In the literature, the most popular test problems for VRPTW are known as Solomon test problems (Cordeau et. al., 2002). These problems appear in three sets. First set is the clustered problems set. Second set consists of problems, which have evenly distributed nodes. The third set is an intersection of the first two. The data belonging to these set of test problems are given in Table 6.5, 6.6 and 6.7 respectively.

Table 6.5 Data belonging to C series problems

Problem	Number of Nodes	Number of Vehicles	Capacity of Vehicles	Due Date	Problem	Number of Nodes	Number of Vehicles	Capacity of Vehicles	Due Date
C101-25	25	10	200	1236	C201-25	25	10	700	3390
C101-50	50	15	200	1236	C201-50	50	15	700	3390
C101-	100	25	200	1236	C201-100	100	25	700	3390
C102-25	25	10	200	1236	C202-25	25	10	700	3390
C102-50	50	15	200	1236	C202-50	50	15	700	3390
C102-	100	25	200	1236	C202-100	100	25	700	3390
C103-25	25	10	200	1236	C203-25	25	10	700	3390
C103-50	50	15	200	1236	C203-50	50	15	700	3390
C103-	100	25	200	1236	C203-100	100	25	700	3390
C104-25	25	10	200	1236	C204-25	25	10	700	3390
C104-50	50	15	200	1236	C204-50	50	15	700	3390
C104-	100	25	200	1236	C204-100	100	25	700	3390
C105-25	25	10	200	1236	C205-25	25	10	700	3390
C105-50	50	15	200	1236	C205-50	50	15	700	3390
C105-	100	25	200	1236	C205-100	100	25	700	3390
C106-25	25	10	200	1236	C206-25	25	10	700	3390
C106-50	50	15	200	1236	C206-50	50	15	700	3390
C106-	100	25	200	1236	C206-100	100	25	700	3390
C107-25	25	10	200	1236	C207-25	25	10	700	3390
C107-50	50	15	200	1236	C207-50	50	15	700	3390
C107-	100	25	200	1236	C207-100	100	25	700	3390
C108-25	25	10	200	1236	C208-25	25	10	700	3390
C108-50	50	15	200	1236	C208-50	50	15	700	3390
C108-	100	25	200	1236	C208-100	100	25	700	3390
C109-25	25	10	200	1236					
C109-50	50	15	200	1236					
C109-	100	25	200	1236					

Table 6.6 Data belonging to R series problems

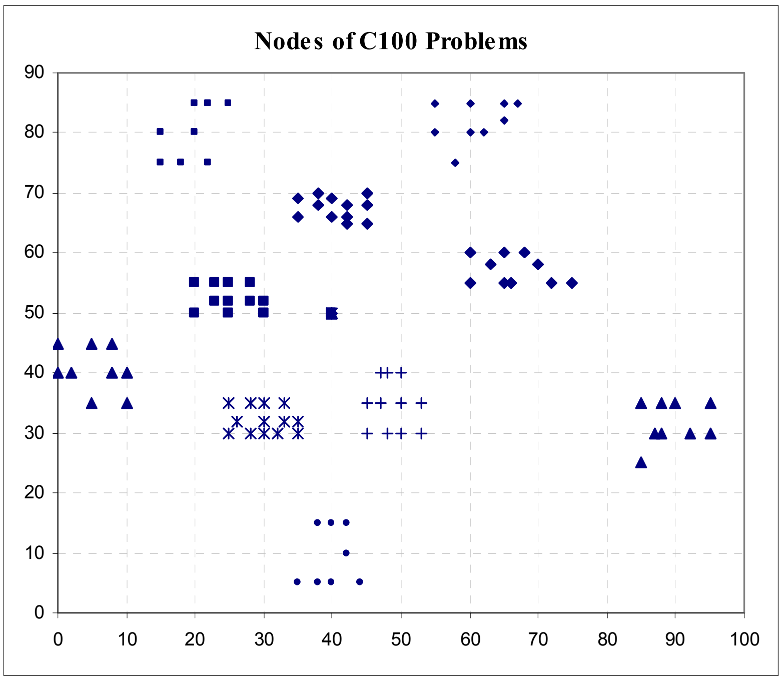
Problem	Number of Nodes	Number of Vehicles	Capacity of Vehicles	Due Date	Problem	Number of Nodes	Number of Vehicles	Capacity of Vehicles	Due Date
R101-25	25	10	200	230	R201-25	25	10	1000	1000
R101-50	50	15	200	230	R201-50	50	15	1000	1000
R101-100	100	25	200	230	R201-100	100	25	1000	1000
R102-25	25	10	200	230	R202-25	25	10	1000	1000
R102-50	50	15	200	230	R202-50	50	15	1000	1000
R102-100	100	25	200	230	R202-100	100	25	1000	1000
R103-25	25	10	200	230	R203-25	25	10	1000	1000
R103-50	50	15	200	230	R203-50	50	15	1000	1000
R103-100	100	25	200	230	R203-100	100	25	1000	1000
R104-25	25	10	200	230	R204-25	25	10	1000	1000
R104-50	50	15	200	230	R204-50	50	15	1000	1000
R104-100	100	25	200	230	R204-100	100	25	1000	1000
R105-25	25	10	200	230	R205-25	25	10	1000	1000
R105-50	50	15	200	230	R205-50	50	15	1000	1000
R105-100	100	25	200	230	R205-100	100	25	1000	1000
R106-25	25	10	200	230	R206-25	25	10	1000	1000
R106-50	50	15	200	230	R206-50	50	15	1000	1000
R106-100	100	25	200	230	R206-100	100	25	1000	1000
R107-25	25	10	200	230	R207-25	25	10	1000	1000
R107-50	50	15	200	230	R207-50	50	15	1000	1000
R107-100	100	25	200	230	R207-100	100	25	1000	1000
R108-25	25	10	200	230	R208-25	25	10	1000	1000
R108-50	50	15	200	230	R208-50	50	15	1000	1000
R108-100	100	25	200	230	R208-100	100	25	1000	1000
R109-25	25	10	200	230	R209-25	25	10	1000	1000
R109-50	50	15	200	230	R209-50	50	15	1000	1000
R109-100	100	25	200	230	R209-100	100	25	1000	1000
R110-25	25	10	200	230	R210-25	25	10	1000	1000
R110-50	50	15	200	230	R210-50	50	15	1000	1000
R110-100	100	25	200	230	R210-100	100	25	1000	1000
R111-25	25	10	200	230	R211-25	25	10	1000	1000
R111-50	50	15	200	230	R211-50	50	15	1000	1000
R111-100	100	25	200	230	R211-100	100	25	1000	1000
R112-25	25	10	200	230					
R112-50	50	15	200	230					
R112-100	100	25	200	230					

Table 6.7 Data belonging to RC series problems

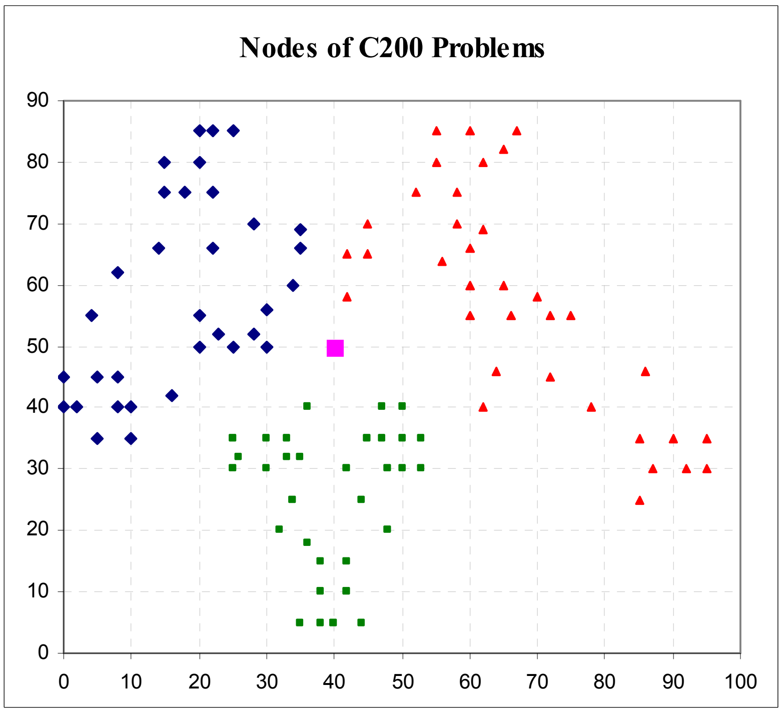
Problem	Number of Nodes	Number of Vehicles	Capacity of Vehicles	Due Date	Problem	Number of Nodes	Number of Vehicles	Capacity of Vehicles	Due Date
RC101-25	25	10	200	240	RC201-25	25	10	1000	960
RC101-50	50	15	200	240	RC201-50	50	15	1000	960
RC101-100	100	25	200	240	RC201-100	100	25	1000	960
RC102-25	25	10	200	240	RC202-25	25	10	1000	960
RC102-50	50	15	200	240	RC202-50	50	15	1000	960
RC102-100	100	25	200	240	RC202-100	100	25	1000	960
RC103-25	25	10	200	240	RC203-25	25	10	1000	960
RC103-50	50	15	200	240	RC203-50	50	15	1000	960
RC103-100	100	25	200	240	RC203-100	100	25	1000	960
RC104-25	25	10	200	240	RC204-25	25	10	1000	960
RC104-50	50	15	200	240	RC204-50	50	15	1000	960
RC104-100	100	25	200	240	RC204-100	100	25	1000	960
RC105-25	25	10	200	240	RC205-25	25	10	1000	960
RC105-50	50	15	200	240	RC205-50	50	15	1000	960
RC105-100	100	25	200	240	RC205-100	100	25	1000	960
RC106-25	25	10	200	240	RC206-25	25	10	1000	960
RC106-50	50	15	200	240	RC206-50	50	15	1000	960
RC106-100	100	25	200	240	RC206-100	100	25	1000	960
RC107-25	25	10	200	240	RC207-25	25	10	1000	960
RC107-50	50	15	200	240	RC207-50	50	15	1000	960
RC107-100	100	25	200	240	RC207-100	100	25	1000	960
RC108-25	25	10	200	240	RC208-25	25	10	1000	960
RC108-50	50	15	200	240	RC208-50	50	15	1000	960
RC108-100	100	25	200	240	RC208-100	100	25	1000	960

6.2.1 Tests on Clustered Problems (C Series)

The locations of nodes in clustered problems can be seen in Figure 6.8.



(a)



(b)

Figure 6.8 The nodes of C100 series can be seen in (a) and C200 series can be seen in (b)

Since the problems in this set are in natural clusters, the routing phase of the Threshold Algorithm (Section 6.1.1) is employed. In other words, each cluster is solved by Model TWM (Formulation 6.1). Computational results can be seen in Tables 6.8 and 6.9. In Table 6.8, the solutions achieved are compared with the approximate optimal solutions in the literature and in Table 6.9, comparison with the best known heuristic solutions are given.

Table 6.8 Performance of the proposed algorithm compared with best known solutions in the literature

Problem No	**Approx. Optimal Solutions**	**Proposed Algorithm**			**Problem No**	**Approx. Optimal Solutions**	**Proposed Algorithm**		
		Cost	**Deviation in Cost**	**Time (sec.)**			**Cost**	**Deviation in Cost**	**Time (sec.)**
C101-25	191,3	191,7	0,23%	<=1	C201-25	214,7	215,5	0,39%	<=1
C101-50	362,4	363,2	0,21%	<=1	C201-50	360,2	361,8	0,44%	118
C101-100	827,3	828,9	0,19%	<=1	C201-100	589,1	591,6	0,42%	281
C102-25	190,3	190,7	0,19%	<=1	C202-25	214,7	215,5	0,39%	46
C102-50	361,4	362,1	0,19%	<=1	C202-50	360,2	361,8	0,44%	166
C102-100	827,3	828,9	0,19%	4	C202-100	589,1	591,6	0,42%	484
C103-25	190,3	190,7	0,19%	<=1	C203-25	214,7	215,5	0,39%	21
C103-50	361,4	362,1	0,19%	<=1	C203-50	359,8	361,8	0,55%	206
C103-100	826,3	828,9	0,31%	27	C203-100	588,7	591,2	0,42%	415
C104-25	186,9	190,7	1,97%	45	C204-25	213,1	215,5	1,13%	183
C104-50	358,0	362,1	1,13%	62	C204-50	350,1	361,8	3,23%	361
C104-100	822,9	828,9	0,72%	89	C204-100	-	591,6		1024
C105-25	191,3	191,7	0,23%	<=1	C205-25	214,7	215,5	0,39%	<=1
C105-50	362,4	363,2	0,21%	<=1	C205-50	359,8	361,8	0,55%	141
C105-100	827,3	828,9	0,19%	<=1	C205-100	586,4	591,6	0,88%	278
C106-25	191,3	191,7	0,23%	<=1	C206-25	214,7	215,5	0,39%	44
C106-50	362,4	363,2	0,21%	<=1	C206-50	359,8	361,8	0,55%	92
C106-100	827,3	828,9	0,19%	<=1	C206-100	586	591,6	0,95%	396
C107-25	191,3	191,7	0,23%	<=1	C207-25	214,5	215,5	0,48%	61
C107-50	362,4	363,2	0,21%	<=1	C207-50	359,6	361,8	0,61%	177
C107-100	827,3	828,9	0,19%	<=1	C207-100	585,8	591,6	0,98%	485
C108-25	191,3	191,7	0,23%	<=1	C208-25	214,5	215,5	0,48%	79
C108-50	362,4	363,2	0,21%	<=1	C208-50	350,5	361,8	3,12%	403
C108-100	827,3	828,9	0,19%	4	C208-100	585,8	591,6	0,98%	561
C109-25	191,3	191,7	0,23%	<=1					
C109-50	362,4	363,2	0,21%	<=1					
C109-100	827,3	828,9	0,19%	9					

The column "*Approximate Optimal Solutions*" in Table 6.8 lists the costs of the optimal solutions found in the literature. However, these solutions were computed with approximate distances obtained by multiplying the real distances by 10 and truncating the result. Hence, the authors state that some routes may not satisfy all time window constraints if real distances were used. Therefore, these solutions are approximate optimal solutions. As seen from Table 6.8, the solutions achieved by the Threshold Algorithm are very close to these values. The average deviation in solution cost from the approximate optimal solutions is 0,55%. This can be interpreted as some of the solutions achieved may be the real optimum though cannot be proved. In addition, since the problems are in clusters, problem sizes are not very large. Therefore, the solution times are very good.

Table 6.9 Performance of the proposed algorithm compared with best known heuristic solutions on C series problems

Problem No	**Number of Nodes**	**Best Known By Heuristics**	**Proposed Algorithm**		
			Cost	**Deviation in Cost**	**Solution Time (sec)**
C101	**100**	**828,9**	**828,9**	**0,00%**	**<=1**
C102	**100**	**828,9**	**828,9**	**0,00%**	**4**
C103	100	828,06	828,9	0,10%	27
C104	100	824,8	828,9	0,49%	89
C105	**100**	**828,9**	**828,9**	**0,00%**	**<=1**
C106	**100**	**828,9**	**828,9**	**0,00%**	**<=1**
C107	**100**	**828,9**	**828,9**	**0,00%**	**<=1**
C108	**100**	**828,9**	**828,9**	**0,00%**	**4**
C109	**100**	**828,9**	**828,9**	**0,00%**	**9**
C201	**100**	**591,6**	**591,6**	**0,00%**	**281**
C202	**100**	**591,6**	**591,6**	**0,00%**	**484**
C203	**100**	**591,2**	**591,2**	**0,00%**	**415**
C204	100	590,6	591,6	0,17%	1024
C205	100	588,9	591,6	0,46%	278
C206	100	588,5	591,6	0,52%	396
C207	100	588,3	591,6	0,56%	485
C208	100	588,3	591,6	0,56%	561

As seen in Table 6.9, in 10 of the 17 problems best known solutions in the literature are achieved by the proposed approach. Also, the solution times for these problems turned out to be very promising. For the other seven problems, very close solutions to the best known are achieved.

The routes achieved by the proposed approach for C series problems can be seen in Appendix I .

6.2.2 Tests on R Series Problems

The locations of nodes in R series problems can be seen in Figure 6.9.

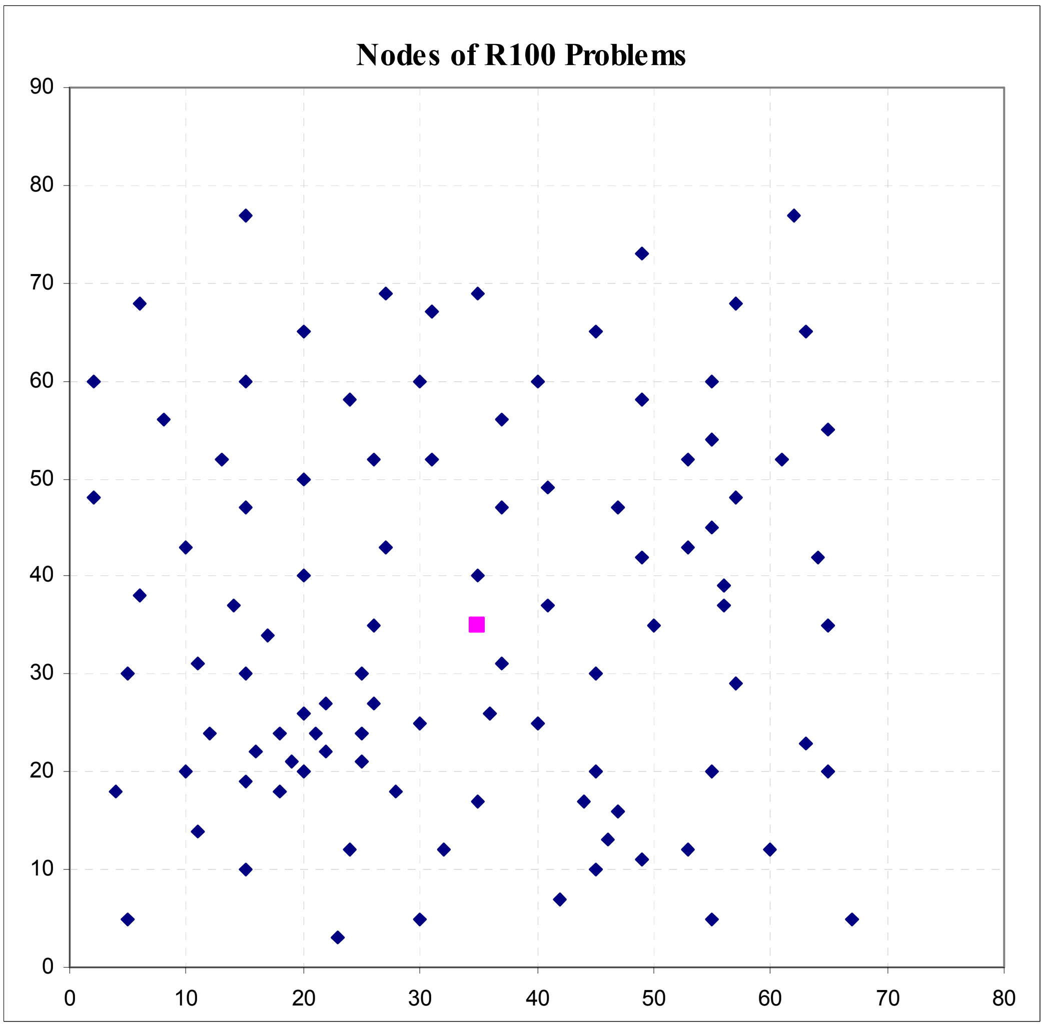


Figure 6.9 The nodes of R series problems

Since these problems cannot be divided into subproblems due to time window inconsistencies, they are solved by the SetCovering Algorithm developed (Section 6.2.2). Computational results can be seen in Table 6.10.

Table 6.10 Performance of the proposed algorithm compared with approximate optimal solutions on R series problems

Problem No	Approximate Optimal Solutions	Proposed Algorithm	
		Cost	Deviation in Cost
R101-25	617,1	618,3	0,20%
R101-50	1044,0	1046,7	0,26%
R101-100	1637,7	1642,9	0,32%
R102-25	547,1	-	
R102-50	909,0	-	
R102-100	1466,6	-	
R103-25	454,6	-	
R103-50	772,9	-	
R103-100	1208,7	-	
R104-25	416,9	-	
R104-50	625,4	-	
R104-100	-	-	
R105-25	530,5	531,5	0,20%
R105-50	899,3	-	
R105-100	1355,3	-	

The SetCovering Algorithm provides the optimum solution when solvable. It can be seen from Table 6.10 that the optimum solutions are provided to R101, 25, 50 and 100 node problems and R105-25 node problem. Also, it is known that the optimal solutions in the literature are achieved with approximate distances. But, with the SetCovering approach proposed in this dissertation, real optimal solutions to the Solomon R101 problems and R105-25 problem are provided. Solution times range between a few minutes and a few hours depending on the problem size. These four solutions provide the real optimums for the literature.

The other problems could not be handled with the proposed approach but to the best of our knowledge, 22 of the R series problems could not be solved with exact approaches in anyway until now. Detailed solutions on R series problems can be seen in Appendix I.

Also, when time windows are tight in a problem, then nearly no subtours exist in the solution. Time window requirements already eliminate subtours since you cannot go back in time. In this case, the number of iterations in the SetCovering algorithm is very small. Hence, it takes negligible time to solve the problem. The procedure turns out to be very beneficial for such problems.

6.2.3 Tests on RC Series Problems

The locations of nodes in RC series problems can be seen in Figure 6.10.

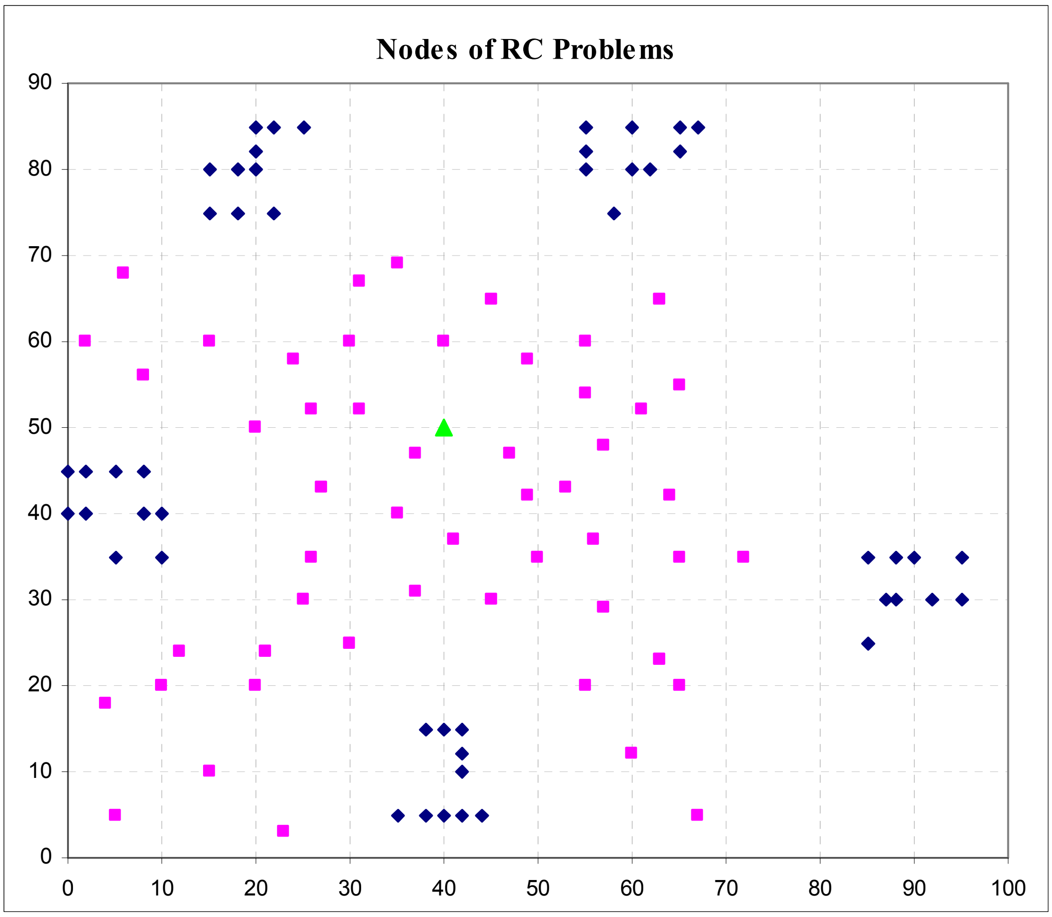


Figure 6.10 The nodes of RC series problems

As stated before, RC series consists of an intersection of the R series and C series nodes. Therefore, this problem can be treated as it is split into six clusters. The smaller size clusters (from Figure 6.10) are solved by Model TWM as the C series problems. The large size problem (from Figure 6.10) is solved by the SetCovering approach as in R series problems. Computational results can be seen in Table 6.11.

Table 6.11 Performance of the proposed algorithm compared with approximate optimal solutions on RC series problems

Problem No	**Optimal Solutions**	**Proposed Algorithm**		**Problem No**	**Optimal Solutions**	**Proposed Algorithm**	
		Cost	**Deviation in Cost**			**Cost**	**Deviation in Cost**
RC101-25	461,1	482,3	4,39%	**RC201-25**	**360,2**	**361,2**	**0,29%**
RC101-50	944,0	968,8	2,56%	RC201-50	684,8	737,3	7,12%
RC101-100	1619,8			RC201-100	1261,8		
RC102-25	**351,8**	**352,9**	**0,32%**	**RC202-25**	**338,0**	**338,8**	**0,24%**
RC102-50	822,5	840,3	2,12%	**RC202-50**	**-**	**615,0**	
RC102-100	1457,4			RC202-100	-		
RC103-25	**332,8**	**334,1**	**0,39%**	**RC203-25**	**326,4**	**327,7**	**0,39%**
RC103-50	710,9	736,3	3,45%	**RC203-50**	**-**	**595,7**	
RC103-100	1258,0			RC203-100	-		
RC104-25	**306,6**	**307,1**	**0,18%**	**RC204-25**	**-**	**300,2**	
RC104-50	545,8	616,9	11,53%	**RC204-50**	**-**	**523,7**	
RC104-100	-			RC204-100	-		
RC105-25	**411,3**	**412,4**	**0,26%**	**RC205-25**	**338,0**	**338,9**	**0,27%**
RC105-50	855,3	890,4	3,94%	**RC205-50**	**631,0**	**632,0**	**0,16%**
RC105-100	1513,7			RC205-100	-		
RC106-25	**345,5**	**346,5**	**0,29%**	**RC206-25**	**324,0**	**325,1**	**0,34%**
RC106-50	**723,2**	**730,4**	**0,98%**	**RC206-50**	**-**	**611,7**	
RC106-100	-			RC206-100	-		
RC107-25	**298,3**	**298,9**	**0,22%**	**RC207-25**	**298,3**	**298,9**	**0,22%**
RC107-50	**642,7**	**645,6**	**0,45%**	**RC207-50**	**-**	**561,4**	
RC107-100	-			RC207-100	-		
RC108-25	**294,5**	**295,0**	**0,17%**	**RC208-25**	**-**	**295,0**	
RC108-50	**598,1**	**599,2**	**0,18%**	**RC208-50**	**-**	**518,5**	
RC108-100	-			RC208-100	-		

Within RC series problems, solutions are provided to 32 out of 48 problems with the proposed approach. For 17 of the 32 solved problems, results achieved have solution costs less within 1% deviation to the approximate optimum solutions in the literature (shown in red in Table 6.11). However, since these optimum solutions are obtained by approximate distances and the results achieved (by the proposed approach) are very close to them, some of these may be the real optimum solutions. But, this cannot be proved because the proposed approach for RC series problems is not a global approach but a partially clustered approach. In addition, from the 32 solved problems, solutions are provided to 8 of the unsolved problems in the literature (shown in bold in Table 6.11). Also, all solution times are within a few seconds.

The solution costs of the other seven solved problems are quite worse than the optimum solutions in the literature. These differences are caused by the difference in the number of vehicles employed. That is, the number of vehicles found by the proposed approach turned out to be more than the number of vehicles found in the literature solutions. Hence, the solution costs exceed the literature solution costs in a recognizable way. But still, the average deviation in solution cost from the approximate optimal solutions is 1,69% which is quite reasonable.

On the other hand 16 of the RC series problems could not be handled with the proposed approach. However, 11 of these problems have remained unsolved in the literature until now. Detailed solutions on RC series problems can be seen in Appendix I.

In Tables 6.12 and 6.13, the summary of the results achieved for these three sets of problems can be seen.

Table 6.12 Comparison of the results in this dissertation and the literature solutions

		Number of Test Problems	Number of Solved	Number of Unsolved
Literature	**C**	51	50	1
Threshold Algorithm		51	51	-
Literature	**R**	69	47	22
Set Covering		69	4	65
Literature	**RC**	48	29	19
Threshold Alg.&Set Covering		48	32	16

Table 6.13 Summary of the results achieved by the proposed approaches

	No. of Solutions to Unsolved Problems	**No. of Optimum Solutions**	**No. of Solutions<1% Cost to Optimum**
C	1	-	45
R	-	4	
RC	8	-	17

In Table 6.12, it is seen that there exist 51 test problems in C series. All 51 test problems in C series are solved by the Threshold Algorithm. However, in the literature, one of these test problems have remained unsolved. That is, among the solutions provided by the Threshold Algorithm, a solution is provided to a problem, which was previously unsolved in the literature. Solution costs and the solution times in this set turned out to be very competitive compared to literature solutions.

In R series problems, there exist 69 test instances (Table 6.12). With the SetCovering Algorithm proposed in this dissertation, only four of them could be solved. However, the solutions to these four problems are global optimum solutions (Table 6.13). In the literature, there exist optimum solutions with approximate distances. Therefore, to best of our knowledge, these four solutions will be recorded in the literature as the new optimum solutions.

Finally, there exist 48 test instances in RC series problems (Table 6.12). Among these instances, eight problems which were unsolved in the literature are solved (Table 6.13) with the proposed procedure in this dissertation. Similar to C series, the solution costs and times are again competitive compared to the literature solutions.

6.3 Conclusion

In this chapter, VRPTW is handled. The threshold algorithm, which was proposed for HVRP previously, is modified according to time window assumptions. But due to time window restrictions, main problem cannot be split into clusters as done in HVRP. Therefore, if the problem has a solvable size or if it is in natural clusters, it is handled with the Threshold Algorithm. However, for other problems a new approach based on set covering logic is proposed. The new approach, which is called the SetCovering Algorithm, runs an IP model, which finds the routes according to time window restrictions. The solution achieved from the IP model is handled by a script algorithm, which looks for subtours and capacity violations in the solution. If there are any, these are included back into the IP model as constraints and the model is rerun. This procedure goes on until no subtours and capacity violations appear in the solution. By this procedure, optimum solution can be achieved.

The popular test problems in the literature, known as Solomon test problems (Cordeau et. al., 2002), are handled with the SetCovering Algorithm (newly designed) and the Threshold Algorithm (previously designed for HVRP and SDVRP). The first set of problems, which is C series problems (clustered problems), are handled by the Threshold Algorithm. Secondly, R series problems (distributed evenly) are considered with the SetCovering Algorithm. Finally, for RC series problems (intersection of R and C series problems), Threshold Algorithm and SetCovering Algorithm are employed partially.

The computational results showed that Threshold Algorithm performs quite well both in terms of solution cost and time for clustered problems. In addition, SetCovering Algorithm achieved some optimum solutions for large scale problems, which are known to be the first optimum solutions in literature.

CHAPTER SEVEN
CONCLUSION

In this chapter, the work carried out throughout the research is summarized and the original contributions, which have appeared within the dissertation are explained.

7.1 Summary

Vehicle Routing Problem (VRP) is concerned with the determination of the optimal routes used by a fleet of vehicles to serve a set of customers. It is one of the hardest combinatorial optimization problems and hence, is one of the most studied among the combinatorial optimization problems, due to both its practical relevance and its considerable difficulty.

Many additional requirements and operational constraints may be imposed on the route construction of the VRP. For example, the service may involve both deliveries and collections, the load along each route must not exceed the given capacity of the vehicles, the total length of each route must not be greater than a prescribed limit, the service of the customers must occur within given time windows, the fleet may contain heterogeneous vehicles, precedence relations may exist between the customers, the customer demands may not be completely known in advance, the service of a customer may be split among different vehicles, and demands or the travel times, may vary dynamically.

The basic version of the problem, known as the capacitated VRP (CVRP), the following assumptions are considered:

- vehicles are homogeneous with known capacities,
- the number of vehicles is unrestricted,
- each customer is visited once and by only one vehicle, and its demand is totally satisfied,
- the customers visited on each route have total demand less than or equal to the capacity of the vehicle assigned to that route,

- each vehicle makes one trip only,
- the objective is to find the set of routes to minimize total distance traveled.

However, many of these assumptions are not realistic for practical applications. For example, the number of vehicles would be limited and the vehicle fleet may be heterogeneous for many of the real life applications. VRP that incorporates heterogeneous vehicle fleet assumption is called heterogeneous VRP (HVRP). Since HVRP is more realistic and has more practical applications, HVRP is one of the challenging problems to be studied.

Another assumption of CVRP, which has an alternative strategy, is that "each customer is visited once and by only one vehicle". In CVRP, split deliveries are not allowed. However, allowing split deliveries may decrease distribution costs. Therefore, split delivery VRP (SDVRP) may provide benefits and should be studied.

In addition to CVRP assumptions, an important variation, which finds application as competition in the markets increase, is the addition of time window requirements of customers. In other words, deliveries are required to be made within certain time windows. When this is the case, the problem is called VRP with time windows (VRPTW).

In the light of the above discussions, HVRP, SDVRP and VRPTW are the basic concerns of this dissertation.

To the best of our knowledge, there is no exact algorithm designed to solve HVRPs to optimum. The proposed algorithm in this dissertation for HVRP is called the Threshold Algorithm. The Threshold Algorithm is a cluster first route second type of algorithm, which integrates advanced mathematical programming tools in each of its phases.

In the clustering phase of the algorithm, in order to work out the tradeoff between number of vehicles and fixed cost of vehicles, an interactive fuzzy goal programming (IFGP) approach is developed. By this approach, HVRP is split into non-intersecting clusters and vehicles are assigned to each cluster to have a number of NP-complete subproblems.

In the routing phase of the algorithm, to construct the routes of each subproblem, a constraint programming (CP) model is built, which works together with a unique subtour elimination algorithm (SEA). SEA is a novel algorithm, which makes complete enumeration among the feasible solutions of the CP model and eliminates the ones, which consist of a subtour and finds the minimum cost solution.

Since there exist no optimum solutions or lower bounds for HVRP in the literature, the newly developed algorithms are tested on benchmark instance from the literature. Therefore, Threshold Algorithm is tested on the 12 instances from Golden et. al. (1984), which are also solved by Taillard (1999), Gendreau et. al. (1999) and Choi and Tcha(2007). The solutions achieved by the Threshold Algorithm are compared with the best known solutions in the literature. Computational results showed that Threshold Algorithm is able to find the best known solutions for some of the test instances. In addition, benefits are provided in terms of computation time for large scale problems. Therefore, for large scale real life HVRPs, Threshold Algorithm may be useful.

Secondly, SDVRP is considered under two cases, which are homogeneous and heterogeneous vehicle fleet. Threshold Algorithm is modified according to the split delivery assumption. For the homogeneous fleet case, vehicle assignment is made by integer programming (IP). For heterogeneous fleet, IFGP approach is employed. In addition, in the routing phase, CP turned out to be unsuccessful for split deliveries. Therefore an IP model is built and SEA is modified to operate with the IP model. Literature problems are again handled with the proposed algorithm and results showed that split deliveries provide beneficial distribution costs.

Finally, VRPTW is handled. Due to some of the special characteristics of time windows, these problems cannot be clustered. Therefore, VRPTW is again considered under two cases as clustered VRPTW and large scale VRPTW. The routing phase of the Threshold Algorithm is modified to work with clustered VRPTW. For large scale VRPTW, a novel SetCovering Algorithm is developed. Similar to previous VRPs, the well known Solomon test problems (Cordeau et. al., 2002) from the literature are solved using the proposed approaches. Some of the problems are solved to optimum, which are known to be the first optimum solutions in the literature.

The summary of the studies within the research can be seen in Figure 7.1

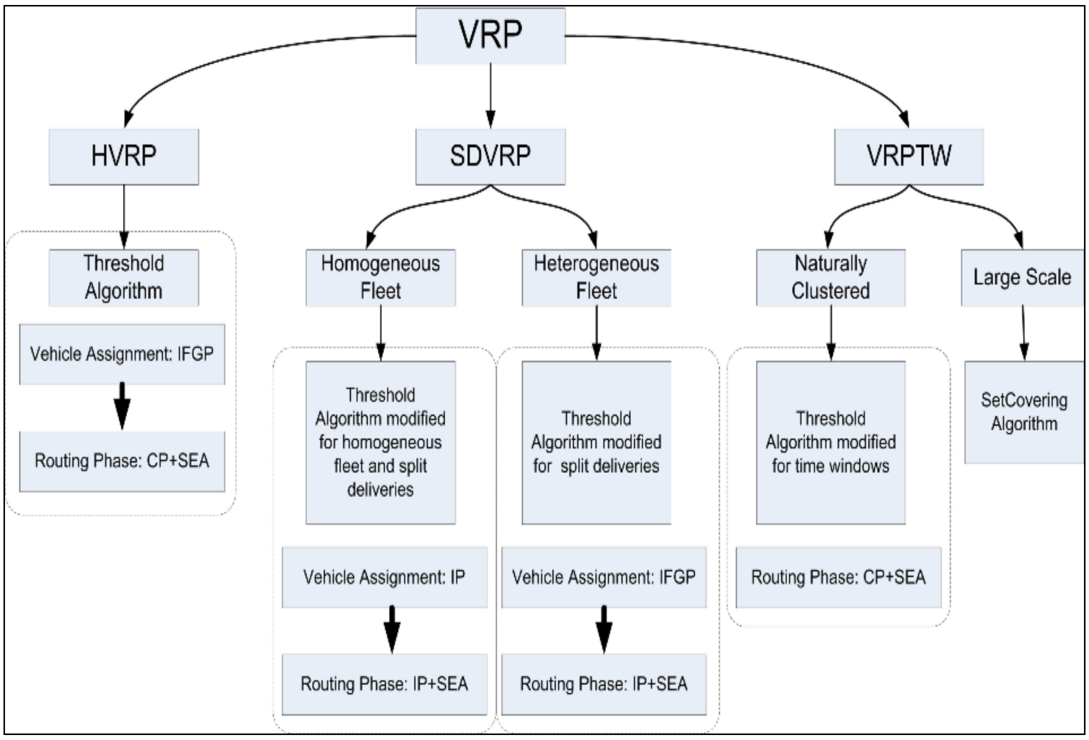


Figure 7.1 Summary of the research carried out in this dissertation

7.2 Original Contributions of the Dissertation

As seen in Figure 7.1, three different variations of VRP are considered in this dissertation, which are HVRP, SDVRP and VRPTW, respectively. In addition to these three variations, HVRP with split deliveries is handled in the dissertation, which to the best we know is the first time that this problem is studied in the literature.

Within the dissertation, a novel Threshold Algorithm is developed and different modifications of this algorithm is employed to solve these variations of VRP. In addition, for large scale VRPTW, which cannot be handled with the Threshold Algorithm due to its special characteristics, an original SetCovering Algorithm is developed. All the methodologies proposed are tested on the well known benchmark instances from the literature and the results are compared with the best known solutions in the literature. Some of the solutions constitute optimum and new best known solutions in the literature.

Additional contributions achieved in this dissertation are listed below.

- In order to deal with the subtour elimination constraints in VRPs, which constitute the main challenge to solve these problems, a novel subtour elimination algorithm (SEA) is proposed. SEA is an iterative algorithm, which operates together with the mathematical model in Threshold Algorithm. Firstly, subtour constraints are completely removed from the mathematical model. Then with SEA, only the subtour constraints, which may appear in the optimum solution, are added back. By this way, the problem can be solved to optimum.

- In HVRPs, there is one more point that makes the problem even more challenging. That is, as the vehicles with different capacities and fixed costs are included in the problem; a tradeoff between total fixed costs and total traveling costs arises. In order to deal with this tradeoff, an IFGP approach is designed in this dissertation. The IFGP approach tries to balance these two objectives by finding the effect of increasing number of vehicles on the fixed cost. To the best of our knowledge, this is the first research, which incorporates fuzzy goal programming into HVRP.

7.3 Future Direction of Research

While this research was conducted, several areas that can be investigated in the future have become clear. Topics worth for future investigation are shown as follows:

- Split delivery assumptions and time window assumptions can be handled together. That is SDVRP and VRPTW can be combined to give SDVRP with time windows. Threshold Algorithm can again be applied to this variation of VRP with necessary modifications. However, there exist no benchmark instances for this problem. Hence, the developed procedure can only be employed for real life cases.

  Similarly, heterogeneous vehicle fleet and time window assumptions can be combined in a single problem to give HVRP with time windows. These two assumption are the two realistic assumptions of VRPs. Therefore, this problem may find many real life applications. Threshold Algorithm can again be applied to these problems.

- Metaheuristic approaches become very popular in the literature for VRPs. The two algorithms proposed in this dissertation, Threshold Algorithm and SetCovering Algorithm may be integrated with metaheuristic approaches. By this way, the fast search provided by metaheuristics and searching for optimum provided by Threshold Algorithm and SetCovering Algorithm can be combined.

  The newly developed procedures may be especially beneficial for VRPTWs. This is because search space can be reduced in a recognizable way due to the special characteristics of time windows.

- There are many other variations of VRPs such as VRP with pickup and delivery, VRP with backhauls, stochastic VRP etc. Threshold Algorithm can be modified according to these variations of VRPs and tested to see its performance.

REFERENCES

Abd El-Wahed, W. F., & Lee, S. M. (2006). Interactive fuzzy goal programming for multi-objective transportation problems. *Omega*, *34* (2), 158-166.

Achuthan, N. R., Caccetta, L., & Hill, S.P. (1997). On the vehicle routing problem. *Proceedings of 2nd World Congress of Nonlinear Analysis*, *30*, 4277-4288.

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

Aminu, U. F., & Eglese, R. W. (2006). A constraint programming approach to the Chinese postman problem with time windows. *Computers & Operations Research, 33*, 3423-3431.

Apt, K. (2003). *Principles of Constraint Programming*. Cambridge, UK: Cambridge University Press.

Archetti C., Speranza M. G., & Hertz A. A. (2006). Tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science, 40(1)*, 64-73.

Archetti C., Savelsbergh M. W. P., & Speranza M.G. (2008). To split or not to split: That is the question. *Transportation Research Part E, 44(1),* 114-123.

Augerat, P., Belenguer, J. M., Benavent, E., Corberan, A., & Naddef, D. (1998). Seperating capacity constraints in the CVRP using tabu search. *European Journal of Operations Research*, *106*, 546-557.

Azi, N., Gendreau, M., & Potvin, J. Y. (2007). An exact algorithm for a single-vehicle routing problem with time windows and multiple routes, *European Journal of Operational Research, 178,* 755-766.

Badeau, P., Guertin, F., Gendreau, M., Potvin, J. Y., & Taillard, E. (1997). A Parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C, 5(2),* 109-122.

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

Baptista, S., Oliveria, R. C., Zuquete, E. (2002). Discrete Optimization: A period vehicle routing case study. *European Journal of Operations Research*, *139*, 220-229.

Baptiste, P., Le Pape, C., & Nuijten, W. (2003). *Constraint Based Scheduling Applying Constraint Programming to Scheduling Problems*. London: Kluwer.

Barbarosoglu, G. & Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research, 29*, 255-270.

Bartak, R. (1999). Constraint Programming: In Pursuit of the Holy Grail. *Proceedings of the Week of Doctoral Students (WDS99) Part IV*, 555-564.

Beasley, J. E. (1983). Route first - Cluster second methods for vehicle routing. *Omega*, *11*, 403-408.

Belenguer J. M., Martinez M. C., & Mota E. (2000). A lower bound fort he split delivery vehicle routing problem. *Operations Research, 48(5),* 801-810.

Bellman, R. E., & Zadeh, L. A. (1970). Decision-making in a fuzzy environment. *Management Science*, *17*, 141-164.

Bent, R., & Hentenryck P. V. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research, 33*, 875-893.

Berger, J., & Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 31*, 2037-2053.

Bouthillier, A., & Crainic, T. G. L. (2005). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research, 32*, 1685-1708.

Braysy, O., Hasle, G., & Dullaert, W. (2004). A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research, 159*, 586-605.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research, 89*, 319-328.

Burchett, D., & Campion, E. (2002). Mix fleet vehicle routing problem - An application of tabu search in the grocery delivery industry. *Management Science Honors Project.*

Choi, E., & Tcha, D. W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research, 34(7)*, 2080-2095.

Chow, K. P., & Perett, M. (1997). Airport Counter Allocation using Constraint Logic Programming. *Proceedings of Practical Application of Constraint Technology (PACT97).*

Cordeau, J. F., Desaulniers, G., Desrosiers, J., Solomon, M. M., & Soumis, F. (2002). VRP with Time Windows. In P. Toth, D., Vigo, (Eds.). *The vehicle routing problem* (157-193). Philadelphia, USA: SIAM.

Desrochers, M., & Verhoog, T. W. (1991). A new heuristic for the fleet size and mix vehicle routing problem. *Computers & Operations Research, 18*, 263-274.

Dincbas, M., & Simonis, H. (1991). APACHE - A Constraint Based, Automated Stand Allocation Systems. *Proceedings of Advanced Software Technology in Air Transport (ASTAIR91).*

Doerner, K. F., Gronalt, M., Hartl, R. F., Kiechle, G., & Reimann, M. (2008). Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. *Computers & Operations Research 35(9),* 3034-3048.

Dondo, R., & Cerda, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research, 176,* 1478-1507.

Dror, M., Laporte, G., & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics, 50*, 239-254.

Fabri, A., & Recht, P. (2006). On dynamic pickup and delivery vehicle routing with several time windows and waiting times. *Transportation Research Part B, 40,* 335-350.

Focacci, F., Lamma, E., Mello, P., & Milano, M. (1997). Constraint Logic Programming for the Crew Rostering Problem. *Proceedings of Practical Application of Constraint Technology (PACT97).*

Frizzell, P. W., & Giffin, J. W. (1995). The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers & Operations Research, 22*, 655-667.

Garcia, B. L., Potvin, J. Y. Rousseau, J. M. (1994). A parallel implementation of the Tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research, 21(9),* 1025-1033.

Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, E. D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research, 26*, 1153-1173.

Gendreau, M., Laporte, G., & Potvin, J. Y. (2002). Metaheuristics for the capacitated VRP. In P. Toth, D., Vigo, (Eds.). *The vehicle routing problem* (129-154). Philadelphia, USA: SIAM.

Ghiani, G., & Improta, G. (2000). An efficient transformation of the generalized vehicle routing problem. *European Journal of Operations Research*, *122,* 11-17.

Golden, B. L., Assad, A. A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research, 11*, 49-66.

Golden, B. L., Assad, A. A., & Wasil, E. A. (2002). Routing vehicles in the real world: Applications in the solid waste, beverage, food, dairy, and newspaper industries. In P. Toth, D., Vigo, (Eds.). *The vehicle routing problem* (245-286). Philadelphia, USA: SIAM.

Hashimoto, H., Ibarakib, T., Imahoric, S., & Yagiuraa, M. (2006). The vehicle routing problem with flexible time windows and traveling times. *Discrete Applied Mathematics, 154,* 2271-2290.

Ho, S. C., & Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research, 31*, 1947-1964.

Hollis, B. L., Forbes, M. A., & Douglas, B. E. (2006). Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. *European Journal of Operational Research*, *173(1)*, 133-150.

Homberger, J., & Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research, 162*, 220-238.

Hu, T. L., & Sheu, J. B. (2003). A fuzzy-based customer classification method for demand-responsive logistical distribution operations. *Fuzzy Sets and Systems, 139,* 431-450.

Hsu, C. I., Hung, S. F., & Li, H. C. (2007).Vehicle routing problem with time-windows for perishable food delivery. *Journal of Food Engineering, 80*, 465-475.

ILOG S.A. (2003). *ILOG OPL Studio 3.7 Language Manual*, France.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research, 33(7),* 2307-2330.

Kallehauge, B., Larsen, J., & Madsen, O. B. G. (2006). Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research, 33*, 1464-1487.

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

Kindervater, G. A. P., & Savelsbergh, M. W. P. (1997). Vehicle Routing: Handling Edge Exchanges. In E. H. L. Aarts, J. K. Lenstra (Eds.). *Local search in combinatorial optimization* (337-360). Chichester, UK: John Wiley and Sons.

Jacquet-Lagreze, E. (1998). Hybrid Methods for Large Scale Optimization Problems: an OR Perspective. *Proceedings of Practical Application of Constraint Technology (PACT98).*

Laporte, G., & Semet, F. (2002). Classical heuristics for the capacitated VRP. In P. Toth, D., Vigo, (Eds.). *The vehicle routing problem* (109-128). Philadelphia, USA: SIAM.

Lau, H. C., Sim, M., & Teo, K. M. (2003). Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research, 148*, 559-569.

Lee, C. G., Epelman, M., White, C. C. & Bozer, Y. A. (2002). A Shortest Path Approach to the Multiple-Vehicle Routing Problem with Split Pick-Ups. *17th International Symposium on Mathematical Programming.*

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

Lima, C. M. R. R., Goldbarg, M. C., & Goldbarg, E. F. G. (2004). A memetic algorithm for the heterogeneous fleet vehicle routing problem. *Electronic Notes in Discrete Mathematics*, *18*, 171-176.

Liu, F. H., & Shen, S. Y. (1999). A method for vehicle routing problem with multiple vehicle types and time windows. *Proceedings of National Science Council ROC, 23*, 526-536.

Liu, F. H. F., & Shen, S.Y. (1999). A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operational Research, 118,* 485-504.

Longo, H., Aragao, M. P., & Uchoa, E. (2006). Solving capacitated arc routing problem using a transformation to the CVRP. *Computers & Operations Research*, *33*, 1823-1837.

Mazzeo, S., & Loiseau, I. (2004). An ant colony algorithm for the capacitated vehicle routing. *Electronic Notes in Discrete Mathematics*, *18,* 181-186.

Mester, D., & Braysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research, 32*, 1593-1614.

Mizrak Ozfirat, P., & Ozkarahan, I. (2006). A cluster first route second algorithm for the heterogeneous vehicle routing problem. *Proceedings of the 26th National Conference on Operations Research and Industrial Engineering*, in Turkish.

Mizrak Ozfirat, P., & Ozkarahan, I. (2007). Comparison of Non-Split and Split Delivery Strategies for the Heterogeneous Vehicle Routing Problem. *Journal of Industrial Engineering, 18(4)*, 2-13.

Mizrak Ozfirat, P., & Ozkarahan, I. (2008a). A Heuristic Based on Constraint Programming for the Vehicle Routing Problem with Time Windows. *International Journal of Computers, Information Technology and Engineering, 2(1),* 45-51.

Mizrak Ozfirat, P., & Ozkarahan, I. (2008b). A Set Covering Approach for the Vehicle Routing Problem with Time Windows. *Proceedings of the Symposium on Modern Scientific Methods'08,* in Turkish.

Moghaddam, R. T, Safaei, N., & Gholipour, Y. (2006). A hybrid simulated annealing for capacitated vehicle routing problems with independent route length. *Applied Mathematics and Computation, 176(2),* 445-454.

Moghaddam R. T., Safaei N., Kah M. M. O., & Rabbani M. (2007). A New Capacitated Vehicle Routing Problem with Split Service for Minimizing Fleet Cost by Simulated Annealing. *Journal of The Franklin Institute, 344*, 406-425.

Mohamed, R. H. (1997). The relationship between goal programming and fuzzy programming. *Fuzzy Sets and Systems*, *89*, 215-222.

Naddef, D., & Rinaldi, G. (2002). Branch-and-cut algorithms for the capacitated VRP. In P. Toth, D., Vigo, (Eds.). *The vehicle routing problem* (53-84). Philadelphia, USA: SIAM.

Nelson, M. D., Nygard, K. E., Griffin, J. H., & Shreve, W. E. (1985). Implementation techniques for the vehicle routing problem. *Computers & Operations Research, 12*, 273-283.

Ochi, L. S., Vianna, D. S., Drummond, L. M. A., & Victor, A. O. (1998). A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems, 14,* 285-292.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research, 41,* 421-451.

Perett, M. (1991). Using Constraint Logic Programming Techniques in Container Port Planning. *ICL Technical Journal*, 537-545.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34(8)*, 2403-2435.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*, 1985-2002.

Ralphs, T. K. (2003). Parallel Branch and cut for capacitated vehicle routing. *Parallel Computing, 29*, 607-629.

Rodrigues, M. M., Souza, C. C., Moura, A. V. (2006). Vehicle and crew scheduling for urban bus lines. *European Journal of Operations Research*, *170(3),* 844-862.

Russell, R. A., & Chiang, W. C. (2006). Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research, 169*, 606-622.

Saez, D., Cortes, C., & Nunez, A. (2008). Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers and Operations Research, 35(11)*, 3412-3438.

Sakawa, M. (1993). *Fuzzy sets and interactive multiobjective optimization*. NY: Plenum Press.

Salhi, S., & Sari, M. (1997). A multi level composite heuristic for the multi depot vehicle fleet mix problem. *European Journal of Operations Research*, *103*, 95-112.

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP'98)*, 417-431.

Sheu, J. B. (2007). A hybrid fuzzy-optimization approach to customer grouping-based logistics distribution operations. *Applied Mathematical Modeling, 31*, 1048-1066.

Taillard, E. D. (1999). A Heuristic Column Generation Method For The Heterogeneous Fleet VRP. *RAIRO, 33*, 1-4.

Tan, K. C., Lee, L. H., & Ou, K. (2001). Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence*, *14*, 825-837.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research, 32*, 2309-2327.

Tarantilis, C. D., & Kiranoudis, C. T. (2001). A meta-heuristic algorithm for the efficient distribution of perishable foods. *Journal of Food Engineering, 50*, 1-9.

Tarantilis, C. D., & Kiranoudis, C. T. (2002). Distribution of fresh meat. *Journal of Food Engineering, 51*, 85-91.

Teodorovic, D., & Pavkovic, G. (1996). The fuzzy set theory approach to the vehicle routing problem when demand at nodes is uncertain. *Fuzzy Sets and Systems, 82(3)*, 307-317.

Toth, P., & Vigo, D. (2002). Branch and Bound Algorithms for the Capacitated VRP. In P. Toth, D., Vigo, (Eds.). *The vehicle routing problem* (29-51). Philadelphia, USA: SIAM.

Tung, D.V, & Pinnoi, A. (2000). Vehicle routing-scheduling for waste collection in Hanoi. *European Journal of Operations Research*, *125*, 449-468.

Ulusoy, G. (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operations Research*, *22*, 329-337.

Wang, H. F., & Fu, C. C. (1997). A generalization of fuzzy goal programming with preemptive structure. *Computers & Operations Research*, *24*, 819-828.

Wasner, M., & Zapfel, G. (2004). An integrated multi-depot hub location vehicle routing model for network planning of parcel service. *International Journal of Production Economics, 90*, 403-419.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*, 338-353.

Zheng, Y., & Liu, B. (2006). Fuzzy vehicle routing model with credibility measure and its hybrid intelligent algorithm. *Applied Mathematics and Computation, 176,* 673-683.

Zhong, Y., & Cole, M. H. (2005). A vehicle routing problem with backhauls and time windows: a guided local search solution. *Transportation Research Part E, 41*, 131-144.

Zimmermann, H. J. (1978). Fuzzy programming and linear programming with several objective function. *Fuzzy Sets and Systems*, *1*, 45-55.

APPENDIX A
OPL STUDIO CODES OF THRESHOLD ALGORITHM FOR HETEROGENEOUS VEHICLE ROUTING PROBLEM

A1. CODE OF MODEL SC

```
SheetConnection sheet("C:\prob7.xls");
range Sets 1..30;
range Cities 0..30;
int Matris[Sets,Cities]from SheetRead(sheet,"cover");
int Matris2[Sets,Cities]from SheetRead(sheet,"distance2");
var int X[Sets] in 0..1;

minimize sum(i in Sets) sum(j in Cities) (X[i]*Matris2[i,j]*Matris[i,j])
subject to
{
  forall(j in Cities) sum(i in Sets) (X[i]*Matris[i,j])>=1;
};
```

A2. CODE OF MODEL VA

```
SheetConnection sheet("C:\prob12.xls");
int customers=30;
int numbervehicles=75;
int subsets=2;
range stores 1..customers;
range subproblems 1..subsets;
range vehicles 1..numbervehicles;
int ymatris[subproblems,stores] from SheetRead(sheet,"matris50");
float demand[stores] from SheetRead(sheet,"demand");
int cost[vehicles] from SheetRead(sheet,"cost");
int cap[vehicles]  from SheetRead(sheet,"capacity");
var int y[subproblems,stores] in 0..1;
var int w[subproblems,vehicles] in 0..1;
var float+ lambda;
maximize lambda
subject to
{
 //Fuzzy constraints;
 lambda<=1-(((sum(i in subproblems, v in vehicles) w[i,v])-5)/1);
 lambda<=1-(((sum(i in subproblems, v in vehicles) cost[v]*w[i,v])-1520)/1430);
 lambda<=1;
 //Every store must be assigned to exactly one sub-problem;
 forall(j in stores) sum(i in subproblems) y[i,j]=1;
 //The subproblem to which a store is assigned should be within its neighborhood;
 forall(i in subproblems, j in stores) y[i,j]<=ymatris[i,j];
 //The total demand of a subproblem must be greater than or equal to the total
capacities of vehicles assigned to that problem;
 forall(i in subproblems) (sum(v in vehicles) (cap[v]*w[i,v])>=sum(j in stores)
(y[i,j]*demand[j]));
 //Each vehicle can be allocated to at most one sub-problem;
 forall(v in vehicles) sum(i in subproblems) w[i,v]<=1;  };
```

A3. CODE OF MODEL CPM

```
setting searchStrategy = SBS;
SheetConnection sheet("C:\probb13.xls");
int Demandpoints = 14;
int Noofvehicles=3;
range Stores 1..Demandpoints;
range Total 0..Demandpoints;
range Vehicles 1..Noofvehicles;
float Distance[Total, Total] from SheetRead(sheet,"distance28sub1");
int Demand[Total] from SheetRead(sheet,"demand28sub1");
int Capacityofvehicle[Vehicles] from SheetRead(sheet,"cap2");
var int X[Total,Total] in 0..1;
solve
{
//Each city is visited exactly once;
forall (i in Stores) sum (j in Total) (X[i,j]>0)=1;
forall (j in Stores) sum (i in Total) (X[i,j]>0)=1;
//The depot is visited at least once;
forall (v in Vehicles) sum (i in Total) (X[i,0]=v) = 1;
forall (v in Vehicles) sum (j in Total) (X[0,j]=v) = 1;
//A vehicle leaves the city that it enters;
forall (i in Total) (sum (j in Total) (X[i,j]) - sum (j in Total) (X[j,i]))=0;
//Capacity of vehicles should not be exceeded;
forall(v in Vehicles)
 (sum(i in Total) Demand[i]*(sum(j in Total) (X[i,j]=v))) <=Capacityofvehicle[v];
};
```

A4. SCRIPT CODE OF SUBTOUR ELIMINATION ALGORITHM (SEA) FOR HVRP

```
Model m("cpvrpfull.mod");
int n:=m.Demandpoints;
range Total 0..n;
range Vehicles 1..m.Noofvehicles;
int matris[1..m.Noofvehicles, 1..n+1];
int i:=0;
int say:=0;
float dist:=0;
int k:=1;
int j:=0;
int gecici:=0;
int bb:=0;
float mindist:=maxint;
ofile pinar("pinar.txt") ;
while m.nextSolution() do
{
 forall(v in Vehicles)
  forall(t in 1..n+1)
    matris[v,t]:=0;

  forall(v in Vehicles)
 {i:=0;
  k:=1;
  j:=0;
  repeat
  {if m.X[i,j]=v then {dist:=dist+m.Distance[i,j];
                        say:=say+1;
                        matris[v,k]:=i;
                        i:=j;
                        k:=k+1;
                        j:=0;
                        if i=0 then break;}
                 else j:=j+1; }
until j>n;}

if say=n+m.Noofvehicles then if mindist>dist then
                                    { mindist:=dist;
                                      forall(v in Vehicles)
                                      {forall(t in 1..n+1)
                                       {cout << matris[v,t] << " - ";
                                        pinar << matris[v,t] << " - ";}
                                       cout << endl;
                                       pinar << endl;}
                                       cout << "Total Travelled= " << dist << endl;
                                       cout << "Time= " << m.getTime() << endl;
                                       pinar << "Total Travelled= " << dist << endl;
                                       pinar << "Time= " << m.getTime() << endl; }
say:=0;
dist:=0;}
pinar.close();
```

APPENDIX B
DETAILED SOLUTIONS OF HETEROGENEOUS VEHICLE ROUTING PROBLEM TEST INSTANCES

B1. TEST INSTANCE 3

Threshold Level= 38

Table B1.1: Minimum values of z_1 and z_2 for Instance 3.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z1	3	675
Minimize Z2	18	360

ExpectedLoss= 33

Table B1.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected.

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	3	675	-
1	4	675	0
2	**5**	**600**	**75**
3	6	580	20

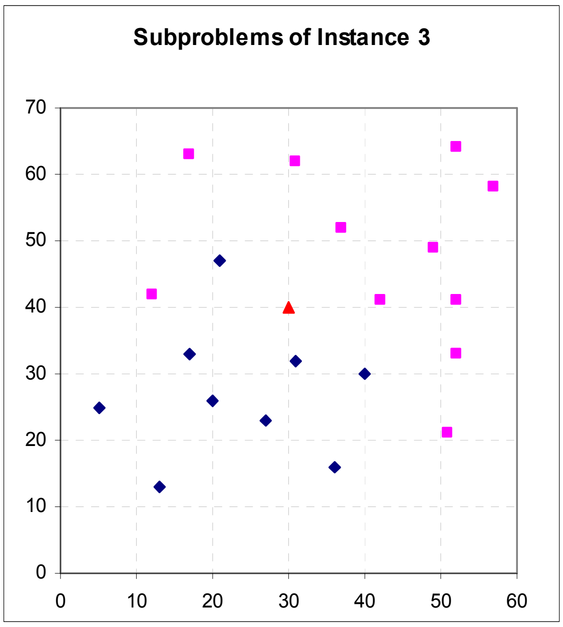


Figure B1.1 Subproblems of Test Instance 3.

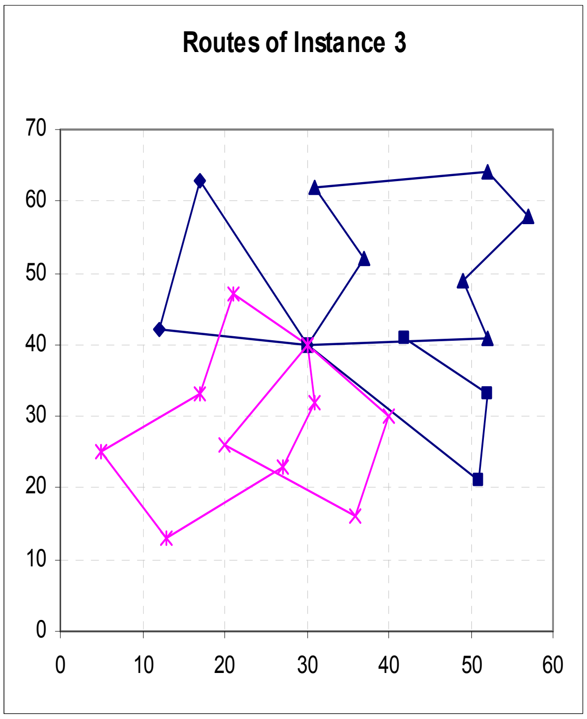


Figure B1.2 Routes of Test Instance 3.

Table B1.3: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Number of Variables X[0..n,0..n] : K	Number of Allowable Values (0,1,….,v) : M	Possible Combinations M^K	Traveling Cost	Computation Time	Route
9	2	100	3	3^{100}	154	15,23	0-4-15-5-0 0-6-18-13-19-17-12-0
11	3	144	4	4^{144}	228	**51,56**	0-16-2-20-3-8-1-0 0-7-14-0 0-10-9-11-0

B2. TEST INSTANCE 4

Threshold Level= 38

Table B2.1: Minimum values of z_1 and z_2 for Instance 4.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	3	7000
Minimize Z_2	6	6000

ExpectedLoss= 33

Table B2.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	3	7000	-
1	4	7000	0
2	5	6500	500
3	**6**	**6000**	**500**

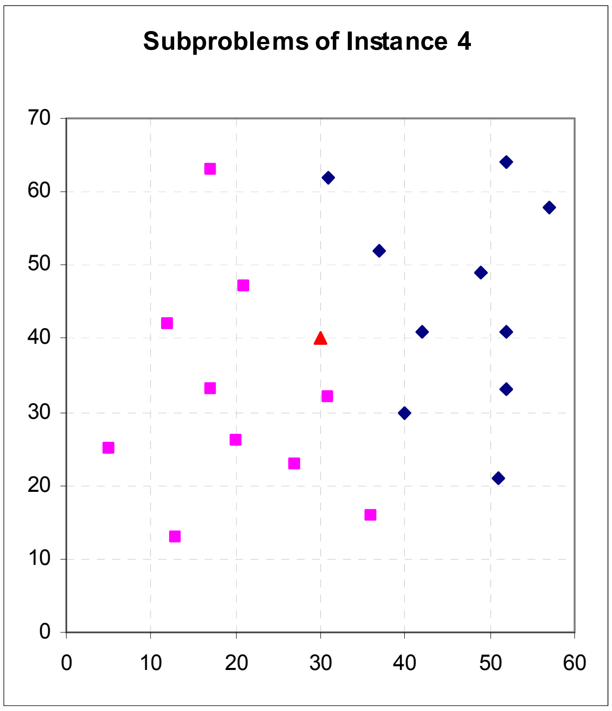


Figure B2.1 Subproblems of Test Instance 4.

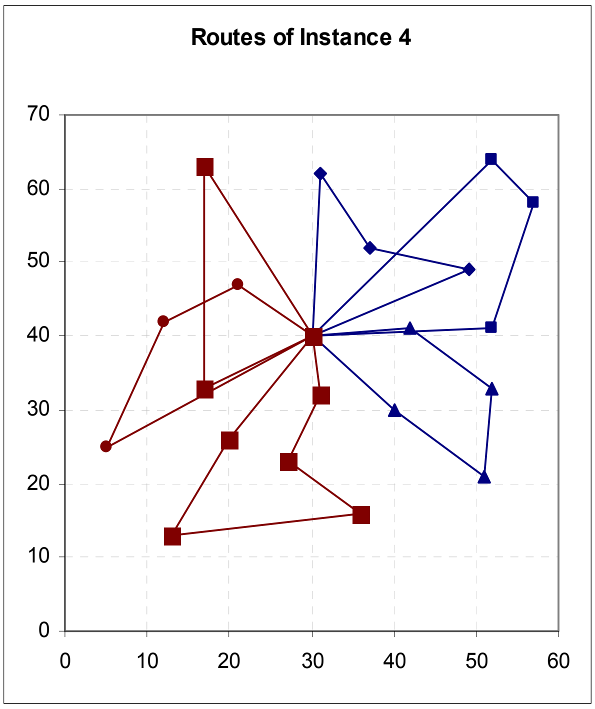


Figure B2.2 Routes of Test Instance 4.

Table B2.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,.....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
10	3	121	4	4^{121}	212	**32,17**	0-8-1-2-0 0-16-20-3-0 0-11-9-10-5-0
10	3	121	4	4^{121}	224	12,59	0-4-19-15-17-12-0 0-18-7-0 0-6-14-13-0

B3. TEST INSTANCE 5

Threshold Level= 38

Table B3.1: Minimum values of z_1 and z_2 for Instance 5.

	Number of Vehicles	**Fixed Cost of Vehicles**
Minimize Z_1	3	675
Minimize Z_2	14	400

$\textit{ExpectedLoss}=54$

Table B3.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	**Number of Vehicles**	**Fixed Cost of Vehicles**	**Difference Between The Last Two Iterations**
0	4	675	-
1	**5**	**600**	**75**
2	6	590	10

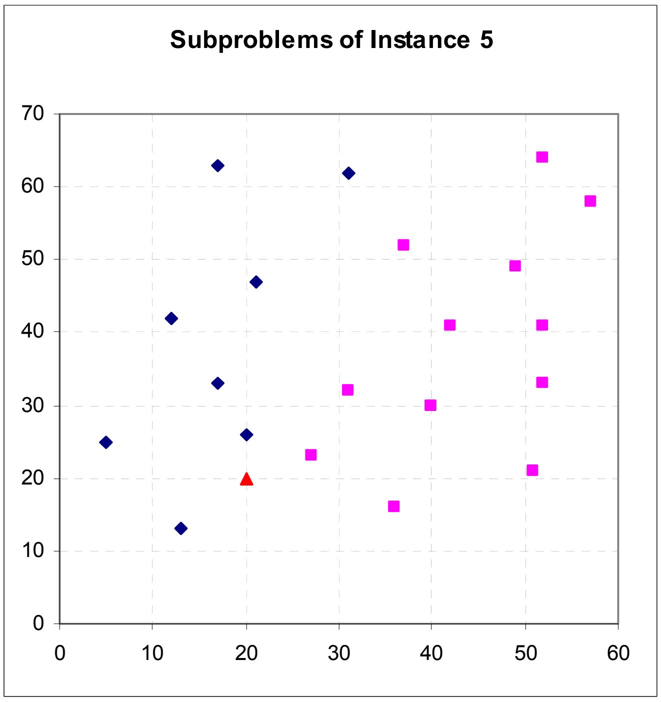


Figure B3.1 Subproblems of Test Instance 5.

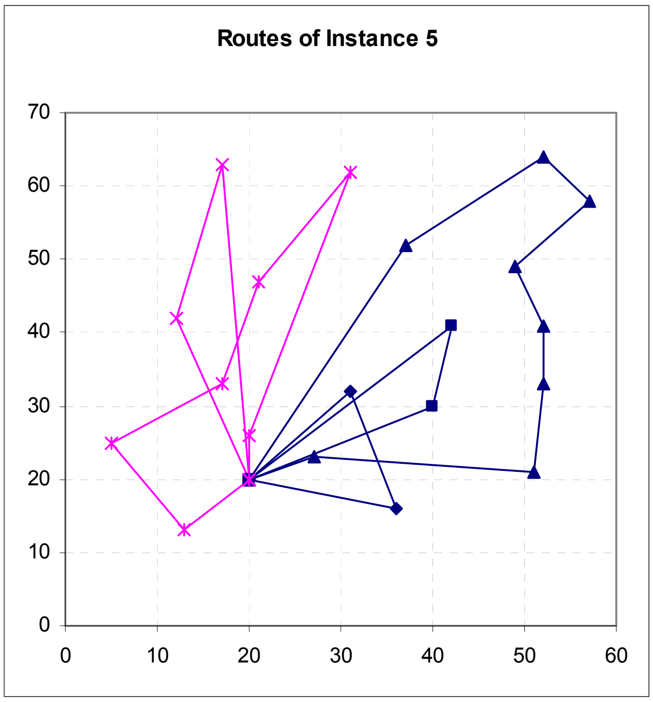


Figure B3.2 Routes of Test Instance 5.

Table B3.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,.....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
8	2	81	3	3^{81}	203,08	2	0-14-7-0 0-4-8-6-18-13-19-0
12	3	169	4	4^{169}	249,07	**101**	0-12-15-0 0-5-11-0 0-17-10-9-16-2-20-3-1-0

B4. TEST INSTANCE 6

Threshold Level= 38

Table B4.1: Minimum values of z_1 and z_2 for Instance 6.

	Number of Vehicles	**Fixed Cost of Vehicles**
Minimize Z_1	3	7000
Minimize Z_2	6	6000

$\textit{ExpectedLoss}=54$

Table B4.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	**Number of Vehicles**	**Fixed Cost of Vehicles**	**Difference Between The Last Two Iterations**
0	3	7000	-
1	4	7000	0
2	5	6500	500
3	**6**	**6000**	**500**

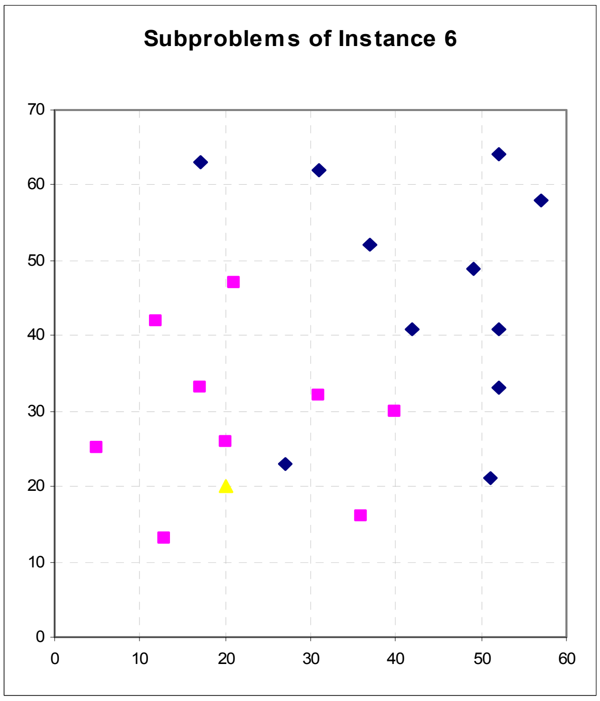


Figure B4.1 Subproblems of Test Instance 6.

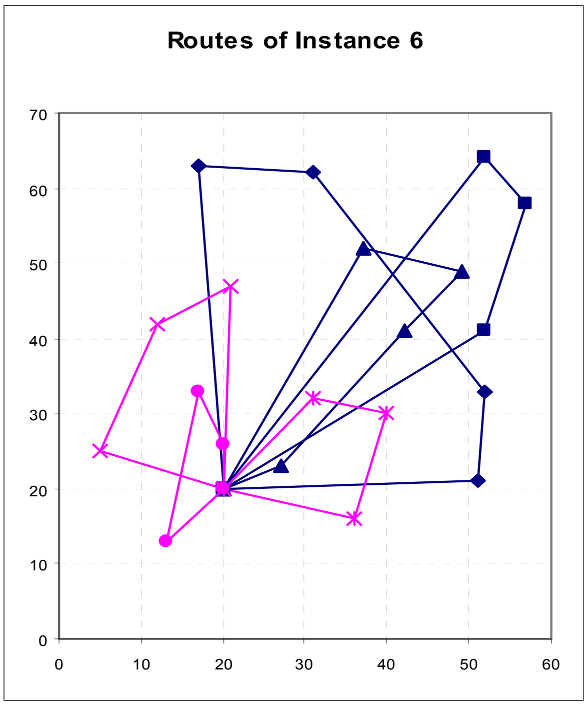


Figure B4.2 Routes of Test Instance 6.

Table B4.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
9	3	100	4	4^{100}	171,97	11	0-13-14-6-0 0-15-5-12-0 0-19-18-4-0
11	3	144	4	4^{144}	344,5	**131**	0-10-9-8-7-0 0-16-20-3-0 0-17-11-2-1-0

B5. TEST INSTANCE 13

Threshold Level= 28

Table B5.1: Minimum values of z_1 and z_2 for Instance 13.

	Number of Vehicles	**Fixed Cost of Vehicles**
Minimize Z_1	6	2225
Minimize Z_2	42	1050

ExpectedLoss=37

Table B5.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	**Number of Vehicles**	**Fixed Cost of Vehicles**	**Difference Between The Last Two Iterations**
0	6	1945	-
1	7	1895	50
2	**8**	**1855**	**40**
3	9	1820	35

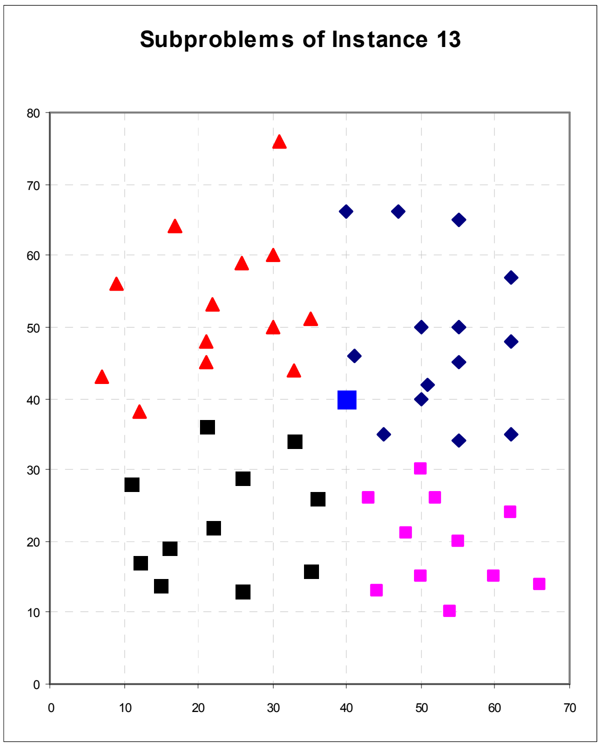


Figure B5.1 Subproblems of Test Instance 13.

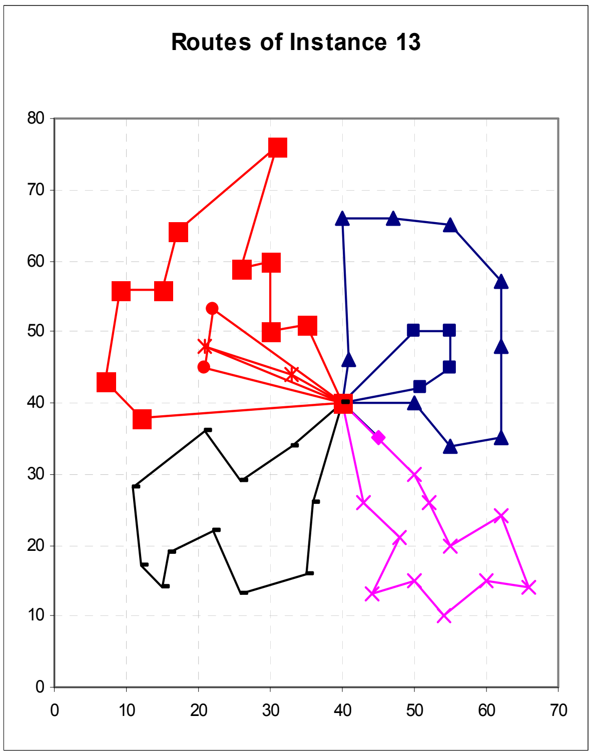


Figure B5.2 Routes of Test Instance 13.

Table B5.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,.....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
11	1	144	2	2^{144}	110,27	21	0-2-28-22-1-43-42-41-23-16-33-6-0
11	1	144	2	2^{144}	101,11	32	0-45-29-5-15-20-37-36-47-21-48-30-0
14	3	225	4	4^{225}	153,15	56	0-4-0 0-46-8-35-7-0 0-26-10-38-11-14-19-13-27-34-0
14	3	225	4	4^{225}	221,25	**112**	0-17-44-0 0-3-32-0 0-49-24-18-50-25-31-9-39-40-12-0

B6. TEST INSTANCE 14

Threshold Level= 30

Table B6.1: Minimum values of z_1 and z_2 for Instance 14.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	6	10000
Minimize Z_2	7	8500

ExpectedLoss=37

Table B6.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	6	10000	-
1	**7**	**8500**	**1500**

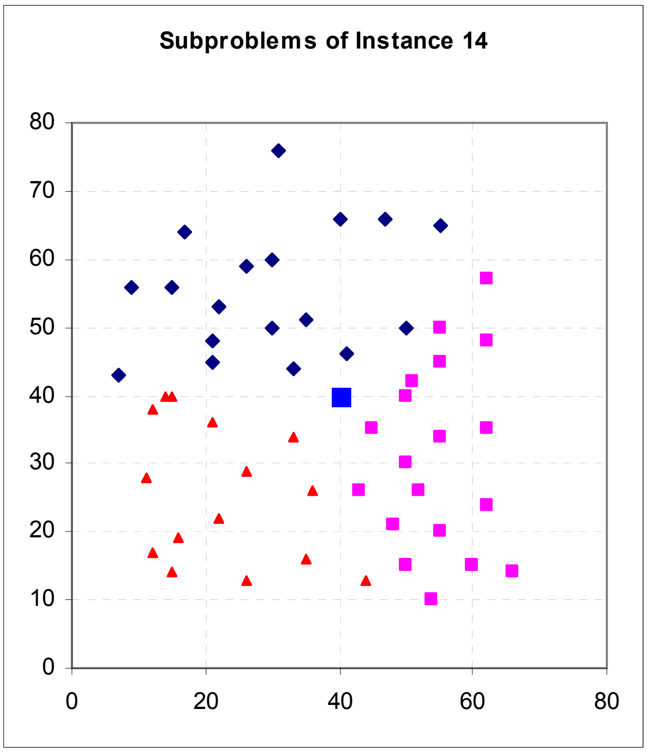


Figure B6.1 Subproblems of Test Instance 14.

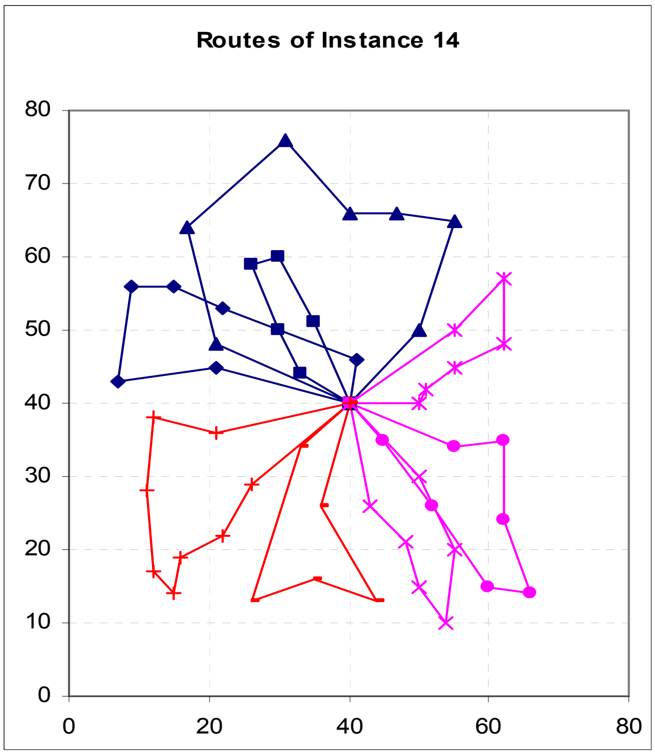


Figure B6.2 Routes of Test Instance 14.

Table B6.3: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Number of Variables X[0..n,0..n] : K	Number of Allowable Values (0,1,.....,v) : M	Possible Combinations M^K	Traveling Cost	Computation Time	Route
13	2	196	3	3^{196}	171,8	87	0-33-1-43-42-41-23-49-16-0 0-6-22-28-21-2-0
18	3	361	4	4^{361}	252,03	233	0-3-24-18-50-32-26-0 0-17-40-9-39-12-0 0-44-25-31-10-38-11-7-0
19	3	400	4	4^{400}	214,42	**373**	0-30-48-47-36-5-45-0 0-34-46-8-19-14-35-0 0-4-29-37-20-15-13-27-0

B7. TEST INSTANCE 15

Threshold Level= 28

Table B7.1: Minimum values of z_1 and z_2 for Instance 15.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	6	2350
Minimize Z_2	16	1600

ExpectedLoss=43

Table B7.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	6	2350	-
1	7	2100	250
2	8	2000	100
3	9	2000	0
4	10	1900	100
5	**11**	**1850**	**50**
6	12	1850	0
7	13	1800	50

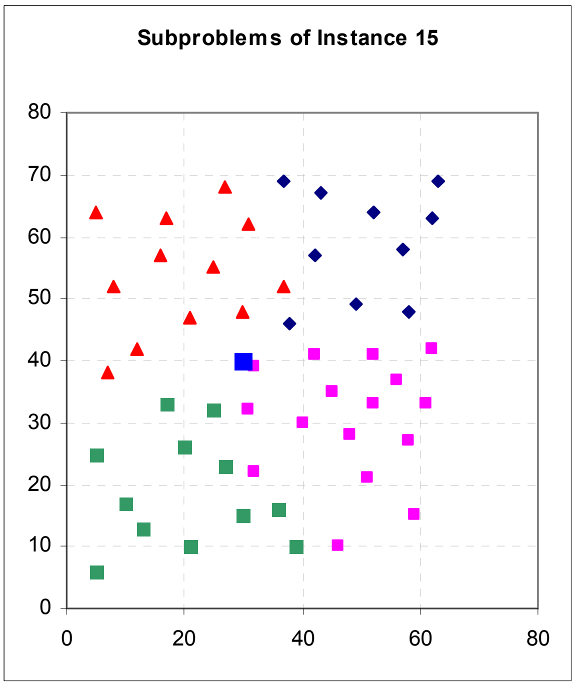


Figure B7.1 Subproblems of Test Instance 15.

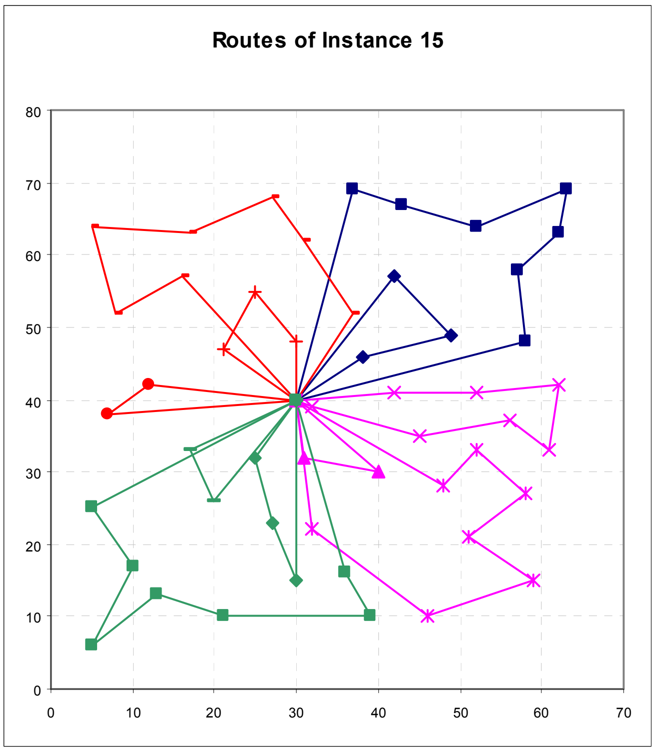


Figure B7.2 Routes of Test Instance 15.

Table B7.2: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Number of Variables X[0..n,0..n] : K	Number of Allowable Values (0,1,.....,v) : M	Possible Combinations M^K	Traveling Cost	Computation Time	Route
10	2	121	3	3^{121}	162,89	20	0-32-2-22-0 0-31-28-3-36-35-20-29-0
12	3	169	4	4^{169}	211,08	36	0-18-4-0 0-44-17-47-0 0-15-45-42-19-40-41-13-0
12	3	169	4	4^{169}	184,36	76	0-25-14-0 0-27-48-6-0 0-23-24-43-7-26-8-1-0
16	3	289	4	4^{289}	212,21	**200**	0-5-12-0 0-11-16-21-34-50-38-46-0 0-49-9-30-10-39-33-37-0

B8. TEST INSTANCE 16

Threshold Level= 34

Table B8.1: Minimum values of z_1 and z_2 for Instance 16.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	7	2600
Minimize Z_2	10	2100

ExpectedLoss=43

Table B8.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	7	2500	-
1	8	2200	300
2	**9**	**2000**	**200**

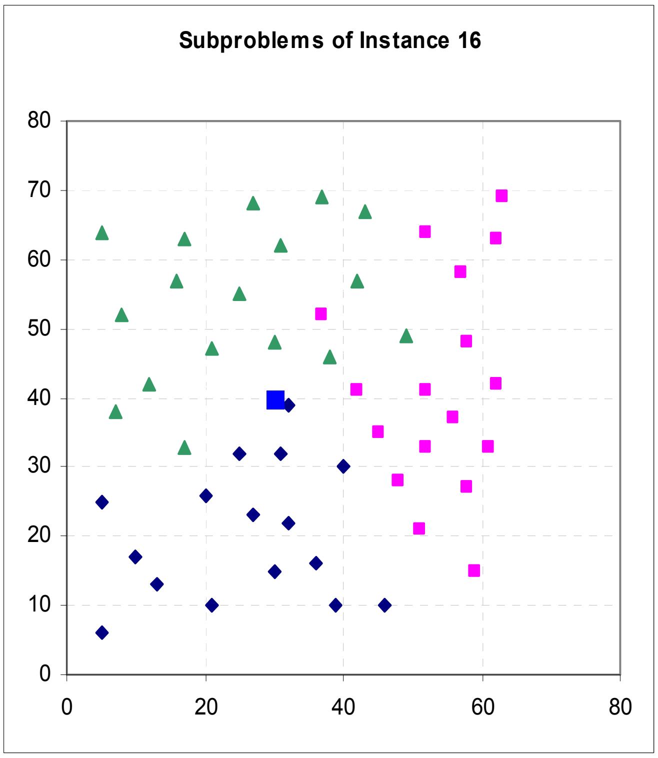


Figure B8.1 Subproblems of Test Instance 16.

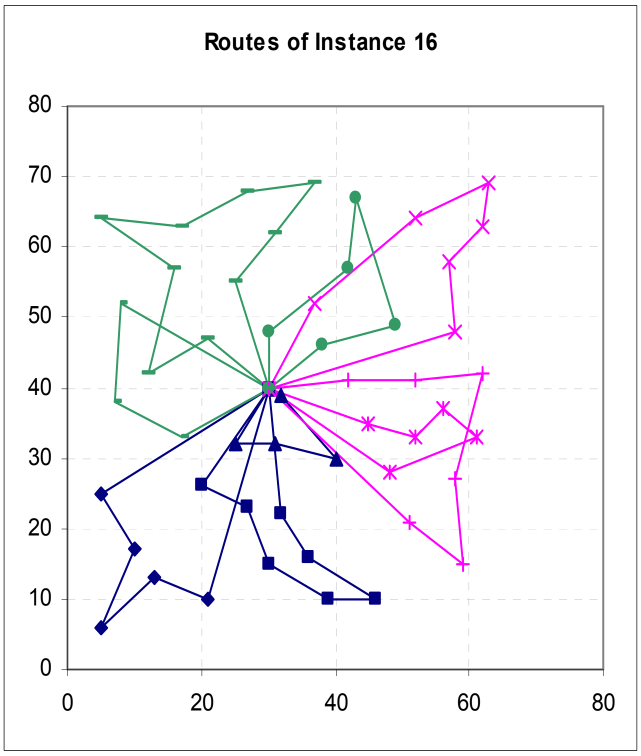


Figure B8.2 Routes of Test Instance 16.

Table B8.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,.....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
16	3	289	4	4^{289}	227,74	118	0-13-41-40-19-42-0 0-4-17-44-45-33-15-37-0 0-47-12-5-46-0
17	3	324	4	4^{324}	252,64	**243**	0-6-25-24-43-23-0 0-18-14-48-0 0-27-7-26-8-31-28-22-2-32-0
17	3	324	4	4^{324}	265,92	197	0-29-20-35-36-3-1-0 0-38-50-34-9-49-0 0-10-39-30-21-16-11-0

B9. TEST INSTANCE 17

Threshold Level= 30

Table B9.1: Minimum values of z_1 and z_2 for Instance 17.

	Number of Vehicles	**Fixed Cost of Vehicles**
Minimize Z_1	5	1260
Minimize Z_2	28	700

ExpectedLoss=43

Table B9.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	**Number of Vehicles**	**Fixed Cost of Vehicles**	**Difference Between The Last Two Iterations**
0	5	1260	
1	6	1190	70
2	7	1080	110
3	**8**	**1005**	**75**
4	9	1000	5

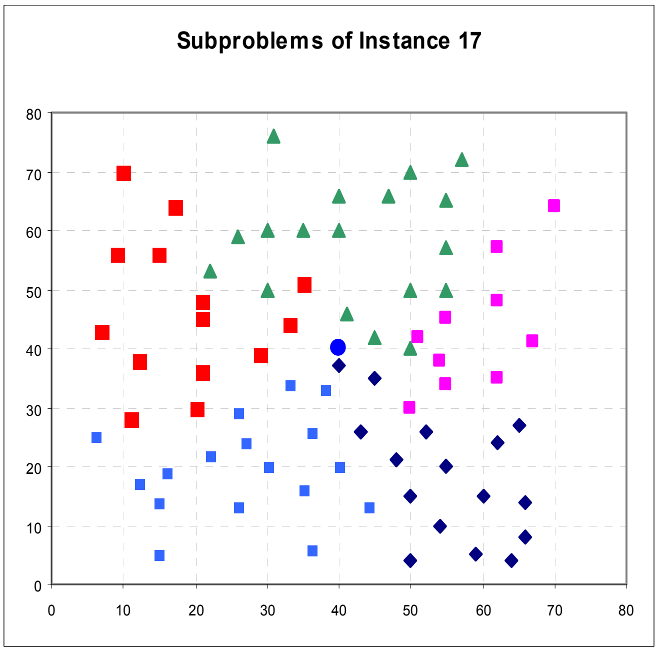


Figure B9.1 Subproblems of Test Instance 17.

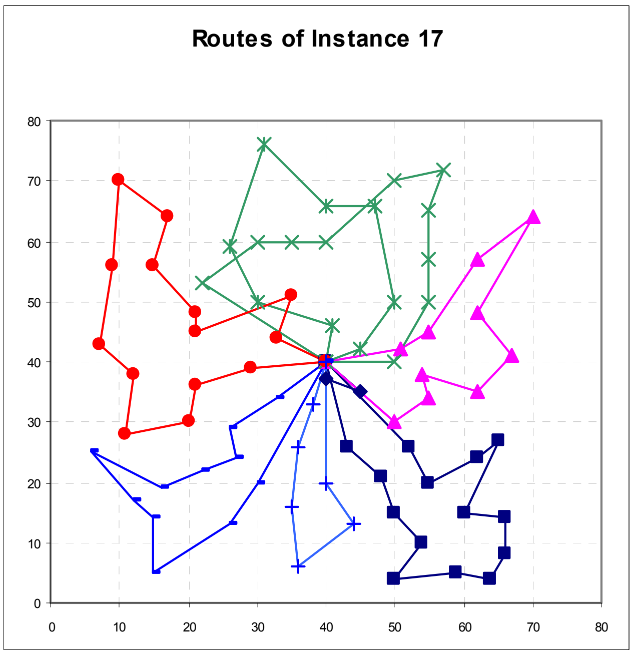


Figure B9.2 Routes of Test Instance 17.

Table B9.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,.....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
10	1	121	2	2^{121}	108,22	19	0-46-8-14-59-19-54-13-52-27-45-0
14	1	225	2	2^{225}	140,24	76	0-51-16-63-23-49-24-18-55-25-50-44-3-12-17-0
16	2	289	3	3^{289}	137,95	110	0-4-75-0 0-29-5-15-57-37-20-70-60-71-69-36-47-48-30-0
17	2	324	3	3^{324}	187,73	114	0-74-21-61-28-2-68-0 0-6-73-62-22-64-42-41-56-43-1-33-0
18	2	361	3	3^{361}	204,62	**340**	0-32-39-72-58-65-66-11-53-35-34-0 0-67-7-38-10-31-9-40-26-0

B10. TEST INSTANCE 18

Threshold Level= 30

Table B10.1: Minimum values of z_1 and z_2 for Instance 18.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	5	2580
Minimize Z_2	49	815

ExpectedLoss=43

Table B10.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	5	2580	-
1	6	2280	300
2	7	2160	120
3	8	2095	65
4	9	2010	85
5	10	1930	80
6	11	1740	190
7	12	1630	110
8	13	1550	80
9	14	1505	45
10	15	1460	45
11	**16**	**1370**	**90**
12	17	1345	25

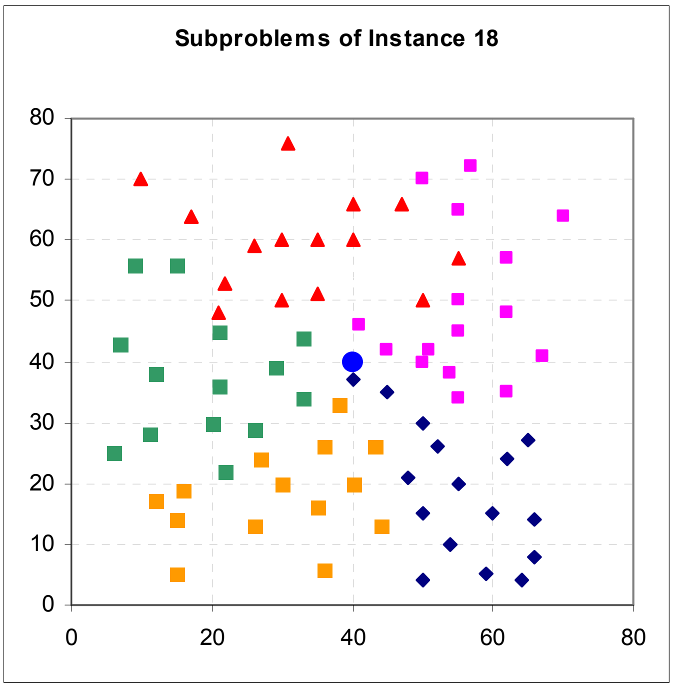


Figure B10.1 Subproblems of Test Instance 18.

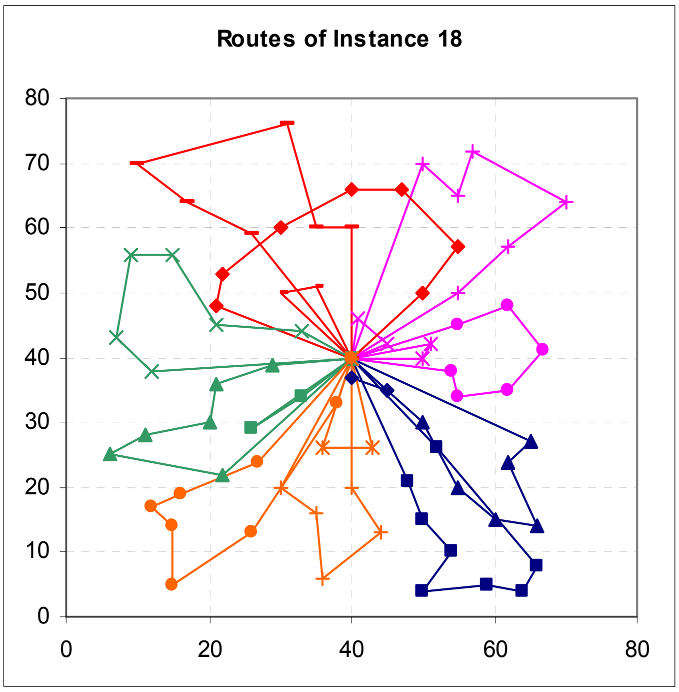


Figure B10.2 Routes of Test Instance 18.

Table B10.3: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Number of Variables X[0..n,0..n] : K**	**Number of Allowable Values (0,1,....,v) : M**	**Possible Combinations M^K**	**Traveling Cost**	**Computation Time**	**Route**
14	3	225	4	4^{225}	208,00	171	0-2-30-0 0-68-22-64-42-41-43-73-0 0-74-21-61-28-62-0
14	3	225	4	4^{225}	205,01	95	0-6-33-0 0-1-56-23-63-16-51-0 0-49-24-18-50-3-17-0
15	3	256	4	4^{256}	227,57	147	0-40-12-0 0-58-72-31-55-25-9-0 0-7-53-38-10-39-32-44-0
16	3	289	4	4^{289}	197,55	442	0-4-75-0 0-29-70-60-71-69-36-47-48-0 0-57-15-20-37-45-0
16	4	289	5	5^{289}	203,05	378	0-26-67-0 0-34-46-0 0-52-27-13-54-19-8-0 0-35-14-59-66-65-11-0

B11. TEST INSTANCE 19

Threshold Level= 38

Table B11.1: Minimum values of z_1 and z_2 for Instance 19.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	8	9100
Minimize Z_2	15	7500

ExpectedLoss=47.

Table B11.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected.

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	8	9100	-
1	9	8900	200
2	10	8700	200
3	11	8500	200
4	12	8300	200
5	13	7900	400
6	14	7700	200
7	**15**	**7500**	**200**

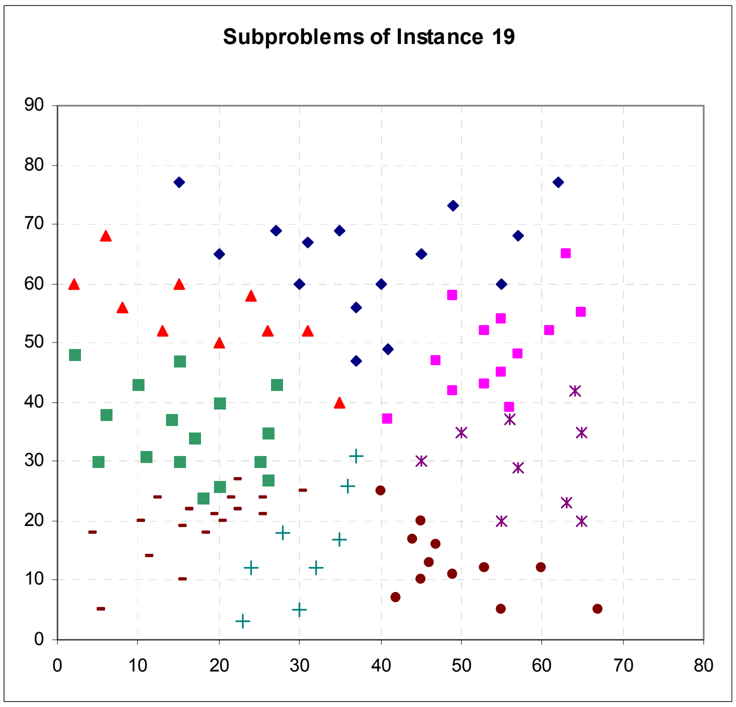


Figure B11.1 Subproblems of Test Instance 19.

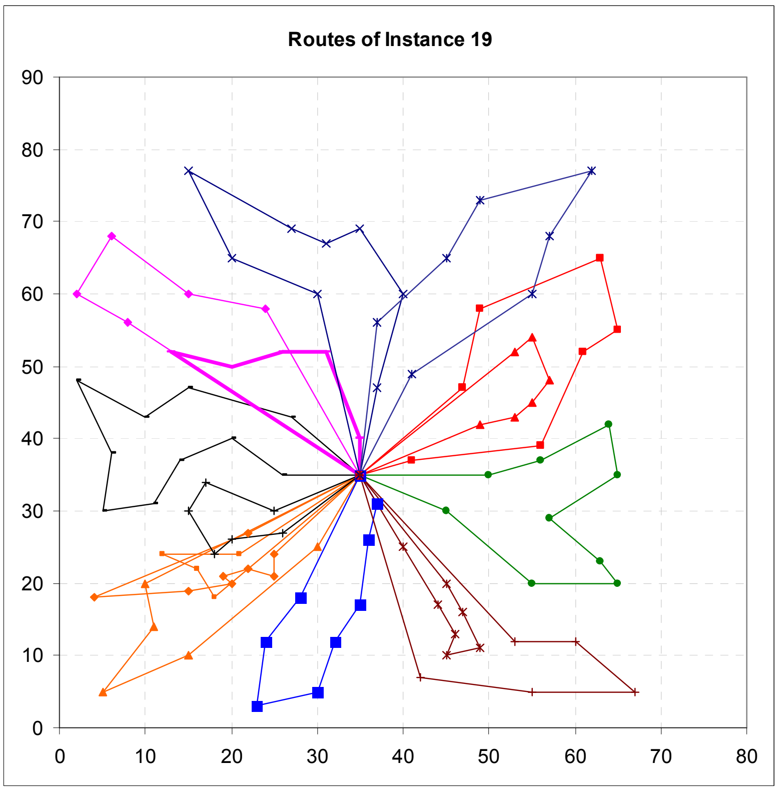


Figure B11.2 Routes of Test Instance 19.

Table B11.3: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Number of Variables X[0..n,0..n] : K	Number of Allowable Values (0,1,.....,v) : M	Possible Combinations M^K	Traveling Cost	Computation Time	Route
15	2	256	3	3^{256}	216,11	131	0-1-9-71-65-66-20-70-0 0-10-11-64-63-90-32-30-69-0
13	2	196	3	3^{196}	154,51	99	0-33-81-79-3-77-76-0 0-28-68-78-34-35-51-50-0
10	2	289	3	3^{289}	161,17	53	0-27-31-88-7-48-0 0-47-36-49-19-62-0
16	2	289	3	3^{289}	147,2	113	0-6-60-5-93-99-94-0 0-52-82-8-46-45-17-84-83-18-
9	1	100	2	2^{100}	95,24	24	0-26-4-25-55-54-24-29-80-12-0
12	2	289	3	3^{289}	159,89	107	0-41-23-67-39-56-0 0-21-72-75-22-74-73-40-0
8	1	81	2	3^{81}	73,67	17	0-53-58-2-57-15-43-42-87-0
17	3	324	4	4^{324}	226,56	453	0-96-86-91-37-98-92-97-95-0 0-100-85-61-59-0 0-13-14-38-44-16-0

B12. TEST INSTANCE 20

Threshold Level= 38

Table B12.1: Minimum values of z_1 and z_2 for Instance 20.

	Number of Vehicles	Fixed Cost of Vehicles
Minimize Z_1	9	3500
Minimize Z_2	25	2500

ExpectedLoss=47.

Table B12.2: Iterations of fuzzy goal programming phase. Iteration in bold is selected

Iteration No	Number of Vehicles	Fixed Cost of Vehicles	Difference Between The Last Two Iterations
0	9	3500	-
1	10	3400	100
2	11	3300	100
3	12	3200	100
4	13	3100	100
5	**14**	**3100**	**-**
6	15	3100	-
7	16	3000	33,3

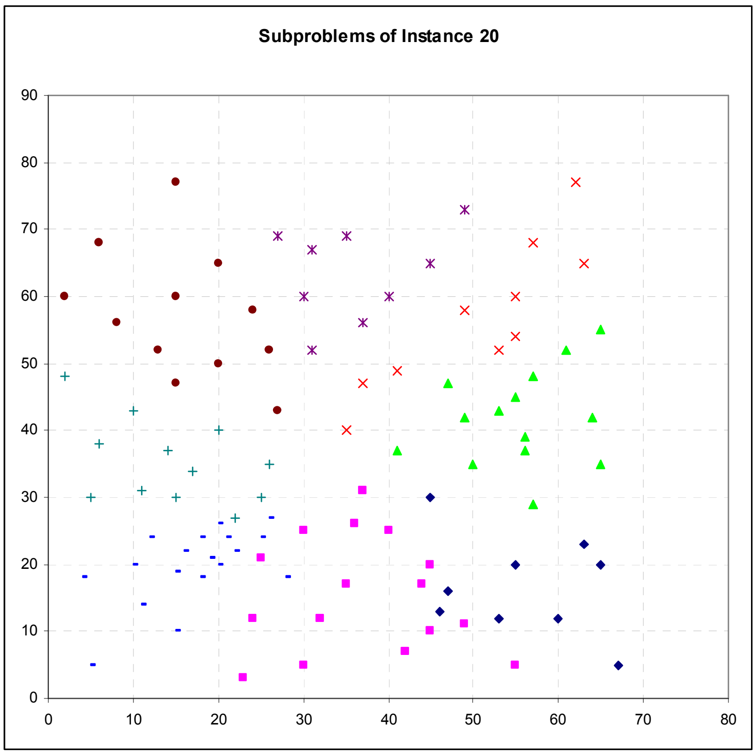


Figure B12.1 Subproblems of Test Instance 20.

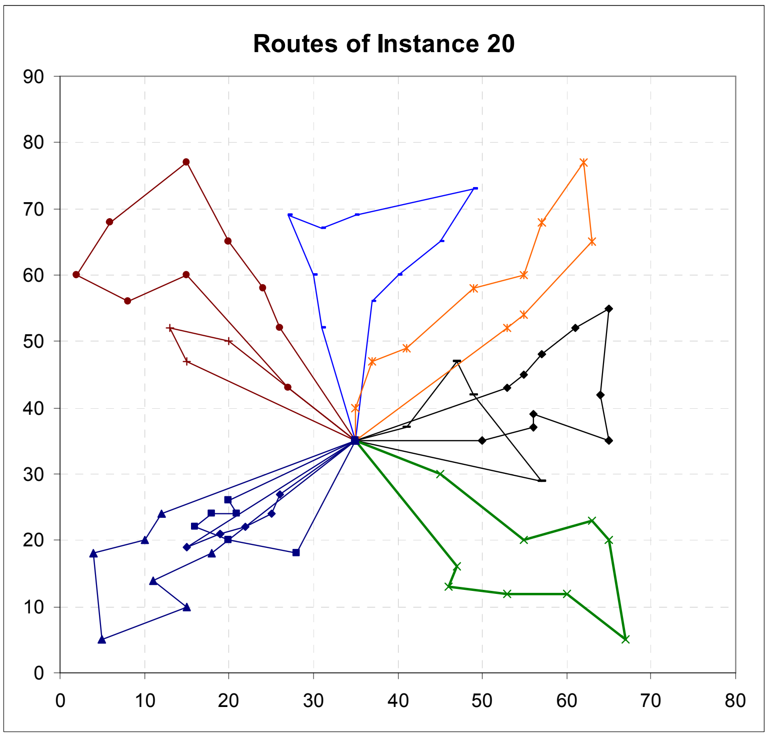
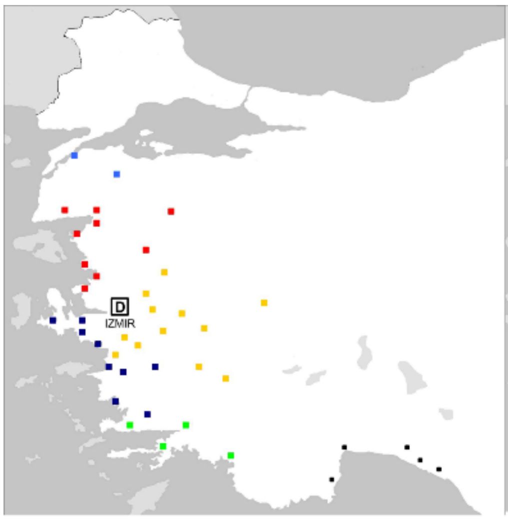


Figure B12.2 Routes of Test Instance 20.

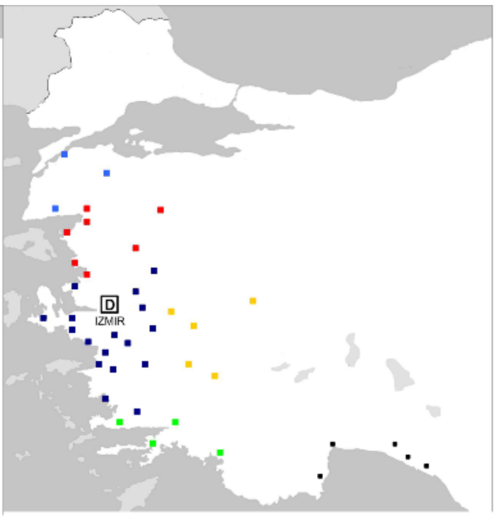
Table B12.3: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Number of Variables X[0..n,0..n] : K	Number of Allowable Values (0,1,....,v) : M	Possible Combinations M^K	Traveling Cost	Computation Time	Route
16	2	289	3	3^{289}	102,2	201	0-13-57-2-40-53-0 0-97-42-43-15-41-22-23-75-73-21-58-0
14	2	225	3	3^{225}	160,29	153	0-54-76-50-28-0 0-77-3-79-78-34-29-24-68-80-12-0
12	2	169	3	3^{169}	151,51	89	0-52-19-47-36-49-64-11-62-88-0 0-7-48-82-0
12	2	169	3	3^{169}	106,90	67	0-6-96-84-17-45-46-8-83-0 0-5-60-18-89-0
9	1	100	2	2^{100}	100,6	5	0-26-4-55-25-67-39-56-74-72-0
9	1	100	2	2^{100}	172,89	11	0-31-10-63-90-32-66-20-30-70-0
10	1	121	2	2^{121}	138,39	23	0-27-69-1-51-9-71-65-35-81-33-0
18	3	361	4	4^{361}	206,99	261	0-94-95-92-98-91-0 0-99-59-93-85-37-87-0 0-100-44-14-38-86-16-61-0

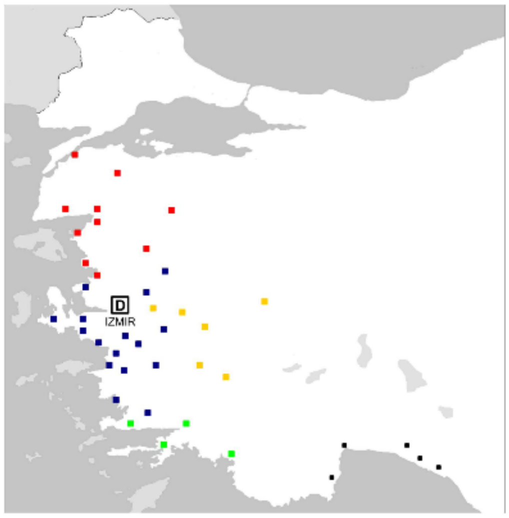
APPENDIX C
SUBPROBLEMS OF DIFFERENT THRESHOLD LEVELS IN RETAIL CHAIN STORE DISTRIBUTION PROBLEM



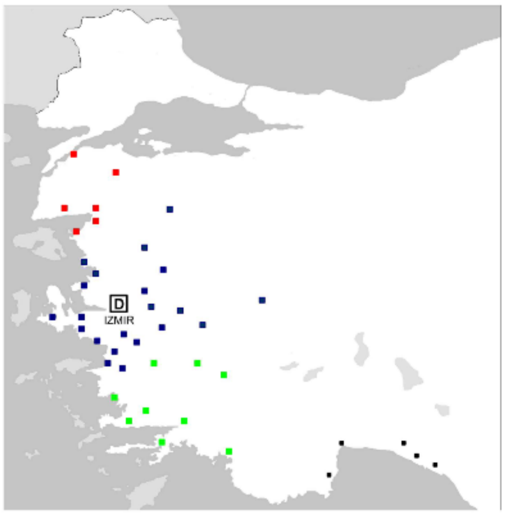
THRESHOLD LEVEL=140



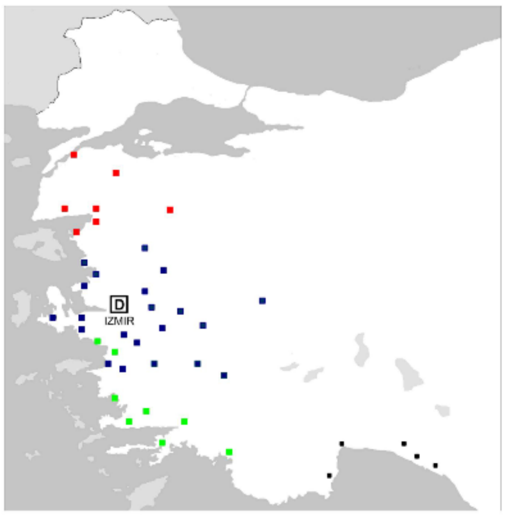
THRESHOLD LEVEL=160



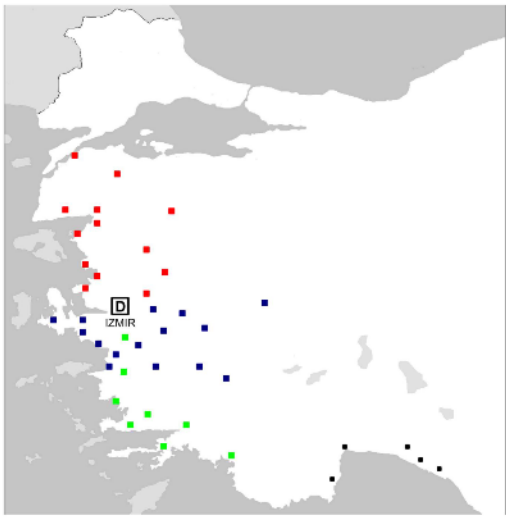
THRESHOLD LEVEL=180



THRESHOLD LEVEL=200



THRESHOLD LEVEL=220



THRESHOLD LEVEL=240

APPENDIX D
OPL STUDIO CODES OF THRESHOLD ALGORITIHM FOR SPLIT DELIVERY VEHICLE ROUTING PROBLEM

D1. CODE OF MODEL SDM

```
setting searchStrategy = SBS;
SheetConnection sheet("C:\archettitest1.xls");

int Demandpoints = 17;
int Noofvehicles=2;

range Stores 1..Demandpoints;
range Total 0..Demandpoints;
range Vehicles 1..Noofvehicles;

float Distance[Total, Total] from SheetRead(sheet,"distance30sub1");
int Demand[Total] from SheetRead(sheet,"demand30sub1");
int Capacity[Vehicles] from SheetRead(sheet,"cap30sub1");

var int X[Total,Total] in 0..2;
var int A[Stores,Vehicles] in 0..28;

solve
{
//Each city can be visited by more than one vehicle (Split deliveries are allowed);
forall(v in Vehicles) forall (i in Stores) sum (j in Total) (X[i,j]=v)<=1;
forall(v in Vehicles) forall (j in Stores) sum (i in Total) (X[i,j]=v)<=1;

//Every vehicle must visit the depot;
forall (v in Vehicles) sum (i in Total) (X[i,0]=v) = 1;
forall (v in Vehicles) sum (j in Total) (X[0,j]=v) = 1;

//A vehicle leaves the city that it enters;
forall(v in Vehicles,i in Total) (sum (j in Total) (X[i,j]) - sum (j in Total) (X[j,i]))=0;

//Capacity constraints;
forall(v in Vehicles) (sum(i in Stores) (A[i,v])-Capacity[v]<=0);

//Demand satisfaction;
forall(i in Stores) sum(v in Vehicles) (A[i,v]) -Demand[i]=0;

//Relation between A and X;
forall(i in Stores,v in Vehicles)(A[i,v]-10000*(sum(j in Total) (X[i,j]=v))<=0);
forall(i in Stores,v in Vehicles) (A[i,v]-(sum(j in Total) (X[i,j]=v))>=0);
};
```

D2. SCRIPT CODE OF SUBTOUR ELIMINATION ALGORITIHM FOR SPLIT DELIVERIES

```
Model m("sdvrp.mod");
int n:=m.Demandpoints;
range Total 0..n;
range Vehicles 1..m.Noofvehicles;
int matris[1..m.Noofvehicles, 1..n+1];
int i:=0;
int say:=0;
float dist:=0;
int k:=1;
int j:=0;
int gecici:=0;
int ek:=0;
int tekrar:=0;
float mindist:=maxint;
ofile pinar("pinar.txt") ;
cout << "merhaba" << endl;

while m.nextSolution() do
{
 forall(v in Vehicles)
  forall(t in 1..n+1)
    matris[v,t]:=0;

 forall(v in Vehicles)
 {i:=0;
  k:=1;
  j:=0;
  repeat
  {if m.X[i,j]=v then {dist:=dist+m.Distance[i,j];
                    say:=say+1;
                    matris[v,k]:=i;
                    i:=j;
                    k:=k+1;
                    j:=0;
                    if i=0 then break;}
               else j:=j+1; }
  until j>n;}

 forall(i in 1..n)
  {forall(v in Vehicles)
   if m.A[i,v]>0 then tekrar:=tekrar+1;
  if tekrar>1 then ek:=ek+tekrar-1;
  tekrar:=0;}

  if say=n+m.Noofvehicles+ek then if mindist>dist then { mindist:=dist;
                                                forall(v in Vehicles)
                                                {forall(t in 1..n+1)
                                                 {cout << matris[v,t] << " - ";
                                                  pinar << matris[v,t] << " - ";}
                                                 cout << endl;
                                                 pinar << endl;}
                                                 cout << "Total Distance Travelled= " << dist <<
endl;
                                                 cout << "Time= " << m.getTime() << endl;
                                                 pinar << "Total Distance Travelled= " << dist <<
endl;
                                                 pinar << "Time= " << m.getTime() << endl; }
  say:=0;
  dist:=0;}
   pinar.close();
```

D3. CODE OF MODEL IPM

```
setting searchStrategy = BFS;
SheetConnection sheet("C:\archettitest3.xls");
int Demandpoints =23;
int Noofvehicles =2;
range Stores 1..Demandpoints;
range Total 0..Demandpoints;
range Boolean 0..1;
range Vehicles 1..Noofvehicles;
float Distance[Total, Total] from SheetRead(sheet,"distancedeneme");
int Demand[Total] from SheetRead(sheet,"demanddeneme");
int Capacityofvehicle[Vehicles] from SheetRead(sheet,"capdeneme");

import int nosubtour;
import int rhs[0..nosubtour];
import int subtours[0..nosubtour,Stores];
range subtourrange 0..nosubtour;

var int X[Total,Total,Vehicles] in 0..1;
var int A[Stores,Vehicles] in 0..41;

minimize  sum(i in Total, j in Total, v in Vehicles) (Distance[i,j]*X[i,j,v])
subject to
{//Each city can be visited by more than one vehicles (Split deliveries are allowed);
forall(v in Vehicles, j in Stores) sum(i in Total) X[i,j,v]<=1;
forall(v in Vehicles, i in Stores) sum(j in Total) X[i,j,v]<=1;

//A vehicle leaves the city that it enters;
forall(k in Total) forall(v in Vehicles)
        ((sum(i in Total) X[i,k,v]) - (sum(j in Total) X[k,j,v]))=0;

//All vehicles should visit the depot node once;
forall(v in Vehicles) sum(j in Stores) X[0,j,v]=1;
forall(v in Vehicles) sum(i in Stores) X[i,0,v]=1;

//Subtour elimination constraints;
forall(t in subtourrange) forall(v in Vehicles)
        sum(i in 1..Demandpoints-1) (X[subtours[t,i],subtours[t,i+1],v])<=rhs[t];
forall(t in subtourrange) forall(v in Vehicles)
        sum(i in 1..Demandpoints-1) (X[subtours[t,i+1],subtours[t,i],v])<=rhs[t];

//Capacity constraints;
forall(v in Vehicles) ((sum(i in Stores) (A[i,v]))-Capacityofvehicle[v]<=0);

//Demand satisfaction;
forall(i in Stores) (sum(v in Vehicles) (A[i,v])) -Demand[i]=0;

//Relation between A and X;
forall(i in Stores, v in Vehicles) A[i,v]- 10000*(sum(j in Total) X[i,j,v])<=0;
forall(i in Stores, v in Vehicles) A[i,v]- (sum(j in Total) X[i,j,v])>=0  ;    };
```

D4. CODE OF SEA FOR SPLIT DELIVERY INTEGER PROGRAMMING MODEL

```
int n:=23;
int Noofvehicles:=2;
range Total 0..n;
range Vehicles 1..Noofvehicles;
range Stores 1..n;
int say:=1;
int saygenel:=1;
int rhsgecici:=0;
int subtour:=0;
int generalsubtour:=0;
int nosubtour:=-1;
int matris[Stores];
int k:=0;
int j:=0;
int i:=1;

Open int subtours[0..nosubtour,Stores];
Open int rhs[0..nosubtour];
ofile pinar("pinar.txt") ;

repeat{
Model main("sdvrpip.mod");
 main.solve();
i:=1;
generalsubtour:=0;
cout << main.objectiveValue() << endl;
pinar << main.objectiveValue() << endl;
 while i<=n do
 { k:=i; say:=1; subtour:=0;
   forall(v in 1..Noofvehicles)
    {j:=0;
     while j<=n do
      {if main.X[k,j,v]=1 then  {matris[say]:=k;
                                k:=j;
                                if j=0 then {say:=1; k:=i;subtour:=0;
                                            break;}
                                if k=matris[1] then {rhsgecici:=say-1;
                                                    say:=say+1;
                                                    while say<=n do
                                                     {matris[say]:=k;
                                                      say:=say+1;}
                                                    subtour:=1;
                                                    break;}
                                j:=-1;
                                say:=say+1;}
    j:=j+1;}
    if subtour=1 then {nosubtour:=nosubtour+1;
                  subtours.addh();
                  rhs.addh();
                  rhs[nosubtour]:=rhsgecici;
                  forall(t in Stores)
                  subtours[nosubtour,t]:=matris[t];
                  generalsubtour:=1; say:=1;}}
                  i:=i+1;}


if generalsubtour=0 then {forall(v in 1..Noofvehicles)
                  {cout << "Vehicle number:"  << v << endl;
                  forall(i in Total)
                  {forall(j in Total)
                     {cout << main.X[i,j,v] << "   " ;
                      pinar << main.X[i,j,v] << "   " ;}
                   cout << endl;
                   pinar << endl;}}
                   cout << main.objectiveValue();
                   pinar << main.objectiveValue();
                   break;}


}until 0;
```

APPENDIX E
DETAILED SOLUTIONS OF DROR AND TRUDEAU (1994) TEST INSTANCES

E1. TEST INSTANCE 1

Threshold Level= 28

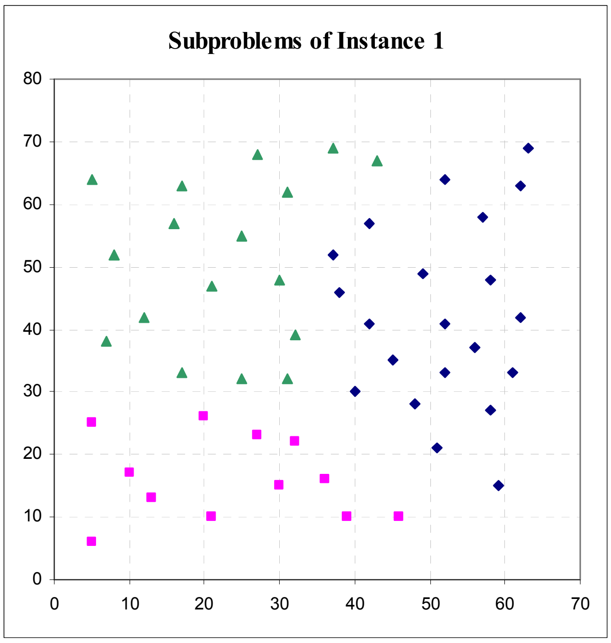


Figure E1.1 Subproblems of Dror and Trudeau test instance 1.

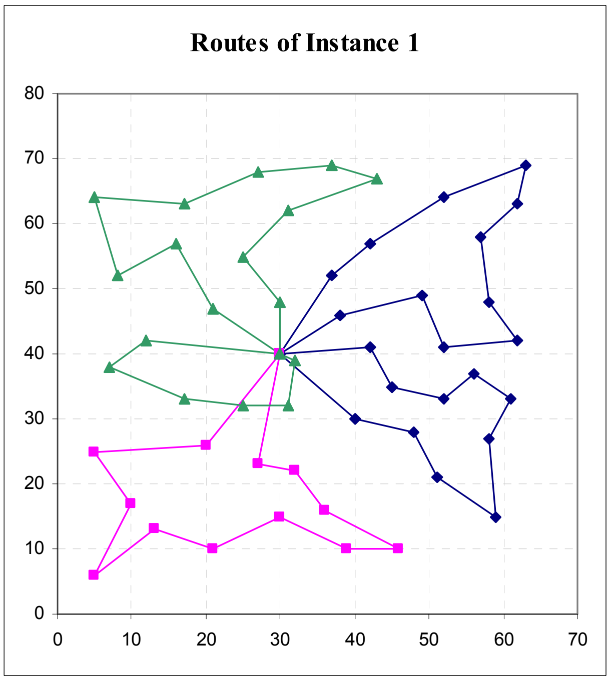


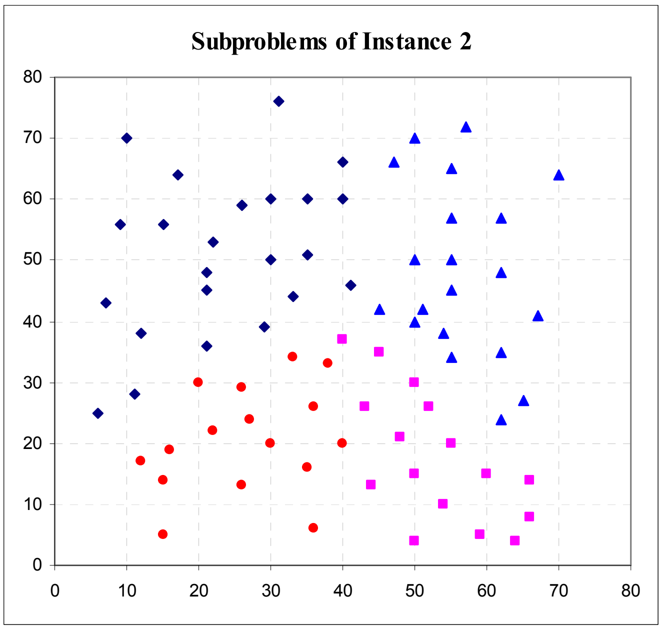
Figure E1.2 Routes of Dror and Trudeau test instance 1.

Table E1.1: Solutions of subproblems.

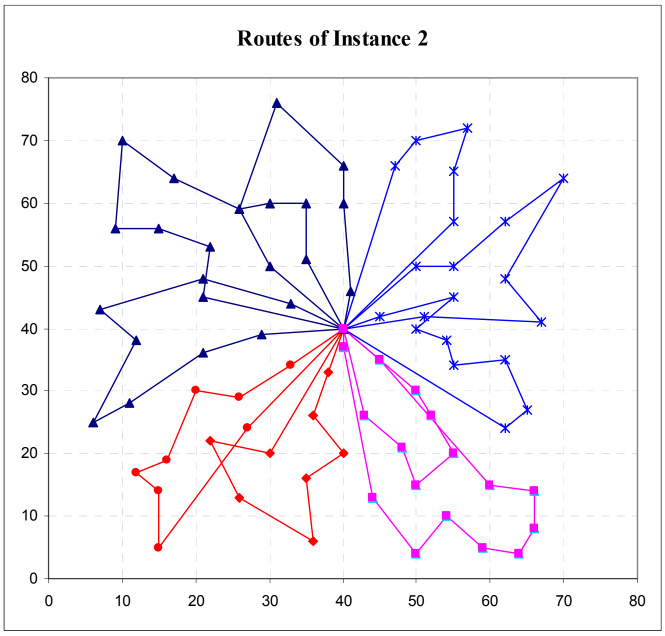
Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
21	2	212,07	71	0-11-38-9-50-34-30-39-10-49-5-0 0-1-22-3-36-35-20-29-21-16-2-32-0
12	2	141,76	11	0-17-37-15-33-45-44-42-19-40-41-13-4-0
17	2	181,87	32	0-14-25-18-47-12-46-0 0-27-48-8-28-31-26-7-43-24-23-6-0

E2. TEST INSTANCE 2

Threshold Level= 30



E1.1 Subproblems of Dror and Trudeau test instance 1.



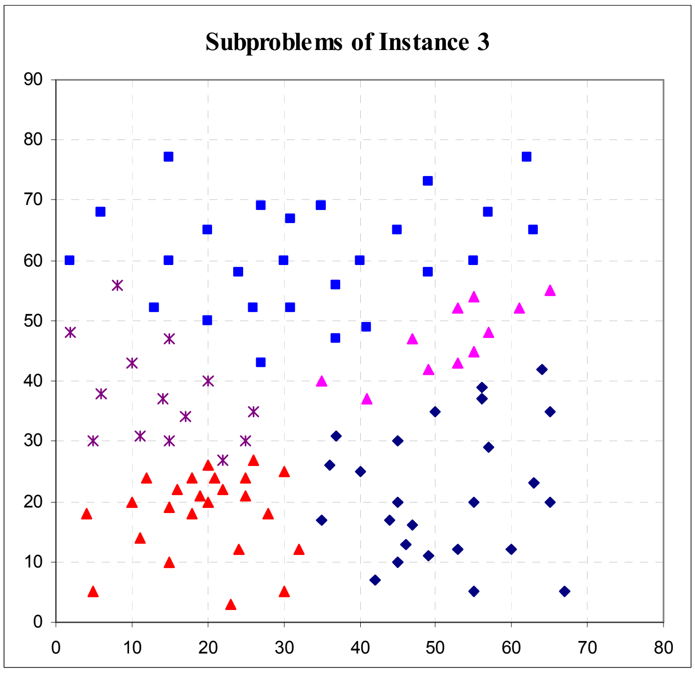
E2.2 Routes of Dror and Trudeau test instance 2.

Table E2.1: Solutions of subproblems.

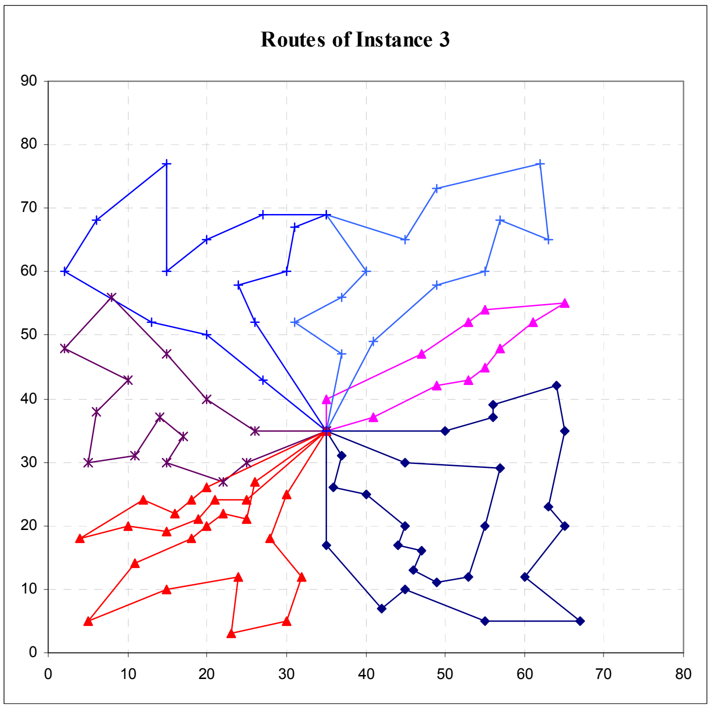
Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
23	3	281,56	501	0-26-58-10-31-39-9-72-12-0 0-3-32-50-18-55-25-9-40-0 0-17-44-24-49-56-23-16-51-0
20	3	249,08	227	0-67-8-34-52-27-13-57-15-0 0-46-54-19-59-14-35-7-0 0-38-65-66-11-53-0
16	2	187,30	164	0-6-33-63-43-41-42-64-73-0 0-68-2-74-28-61-22-1-62-0
16	2	166,26	202	0-4-45-29-5-47-48-30-0 0-4-37-20-70-60-71-36-69-21-75-0

E3. TEST INSTANCE 3

Threshold Level= 32



E1.1 Subproblems of Dror and Trudeau test instance 1.



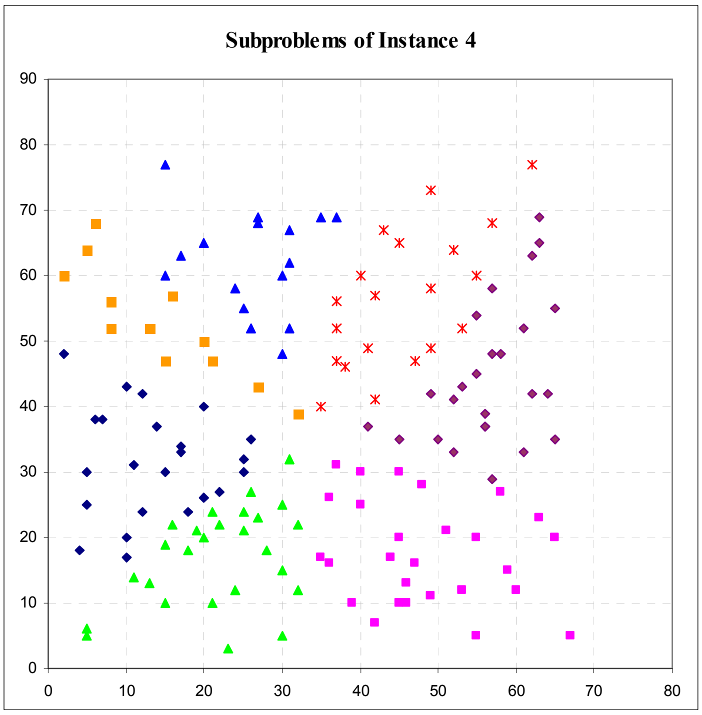
E3.2 Routes of Dror and Trudeau test instance 3.

Table E3.1: Solutions of subproblems.

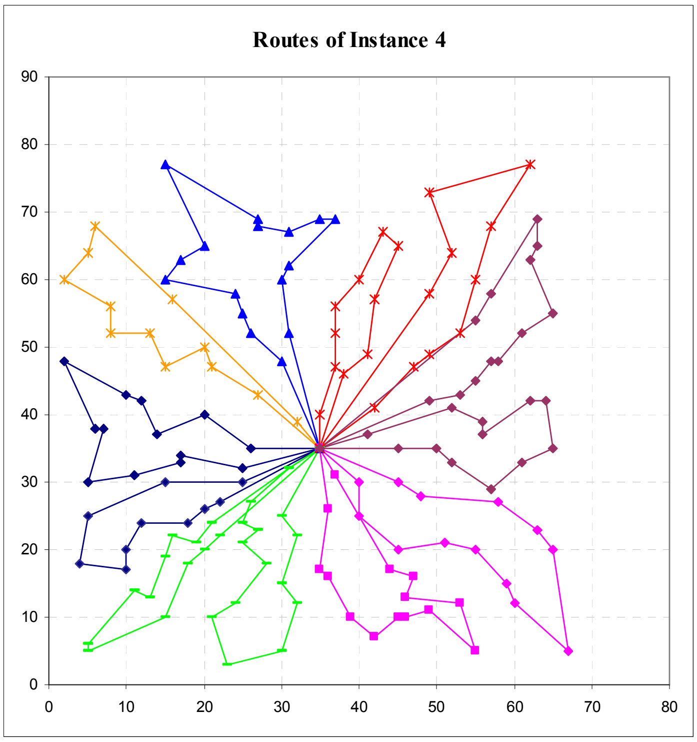
Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
11	1	76,55	54	0-27-50-33-81-34-78-79-3-77-76-28-0
14	1	115,25	189	0-89-18-82-47-46-8-45-17-84-83-60-5-96-6-0
25	2	210,34	554	0-2-41-22--23-67-39-25-55-24-29-68-80-12-0 0-53-58-40-21-73-72-74-75-56-4-54-26-0
24	2	191,50	599	0-95-59-98-91-16-86-61-85-93-99-0 0-94-97-92-37-100-44-38-14-42-43-15-57-87-13-0
26	2	283,72	615	0-69-31-70-30-32-20-66-65-35-71-9-51-1-0 0-52-7-48-36-49-64-19-11-63-32-90-10-62-88-0

E4. TEST INSTANCE 4

Threshold Level= 20



E1.1 Subproblems of Dror and Trudeau test instance 1.



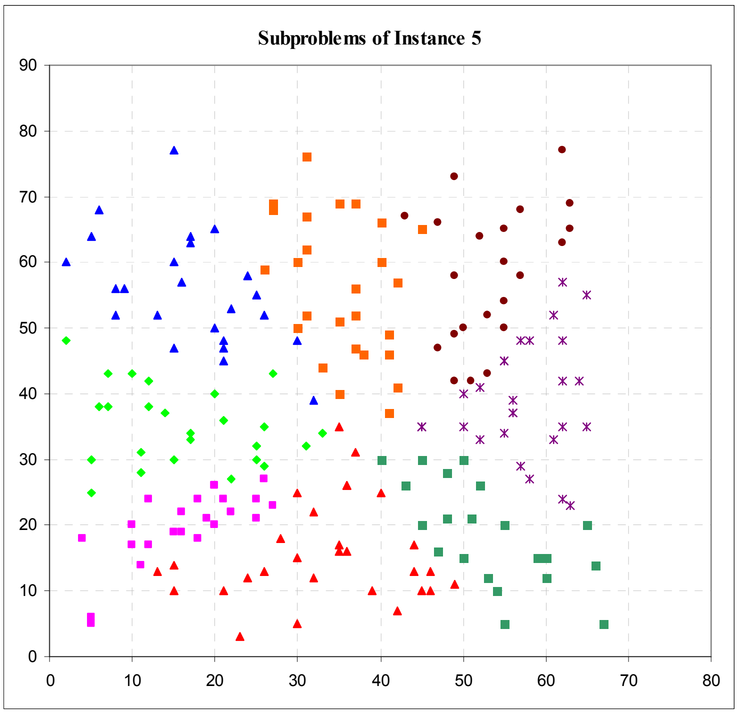
E4.2 Routes of Dror and Trudeau test instance 4.

Table E4.1: Solutions of subproblems.

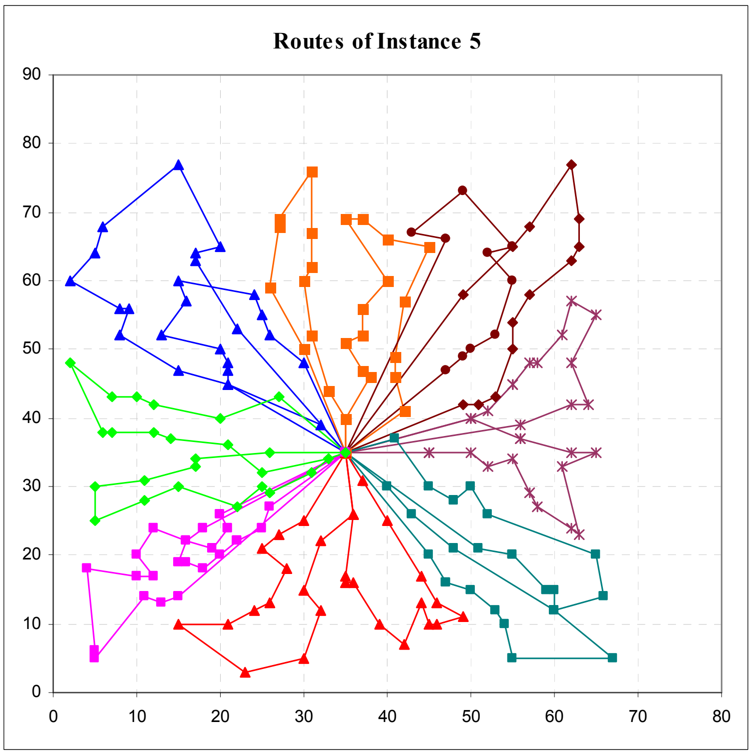
Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
12	1	102,19	58	0-102-6-57-132-98-24-97-86-43-99-23-46-0
16	1	118,34	167	0-27-138-48-112-69-7-61-114-113-26-140-82-31-8-60-81-0
26	2	173,93	325	0-63-37-44-107-65-93-42-92-137-147-17-145-144-12-0 0-142-87-150-64-88-40-94-19-141-135-148-109-12-0
27	2	186,96	407	0-76-49-30-105-75-117-89-39-54-10-71-90-5-0 0-103-123-122-124-106-73-125-33-72-91-45-15-52-108-0
24	2	172,00	290	0-56-55-13-136-41-66-111-143-4-149-146-0 0-139-68-133-14-58-96-95-25-67-134-18-110-47-0
20	2	177,34	272	0-101-3-116-115-121-59-83-2-100-11-0 0-77-119-1-120-80-28-70-22-51-32-0
25	2	172,32	301	0-131-20-36-85-35-84-128-29-129-53-127-126-0 0-38-62-9-104-34-74-79-21-130-50-118-16-78-0

E5. TEST INSTANCE 5

Threshold Level= 30



E1.1 Subproblems of Dror and Trudeau test instance 1.

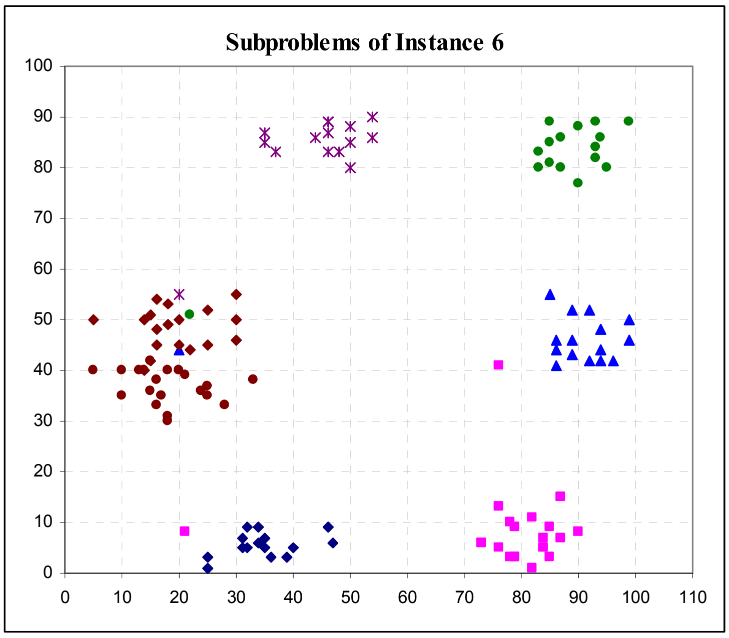


E5.2 Routes of Dror and Trudeau test instance 5.

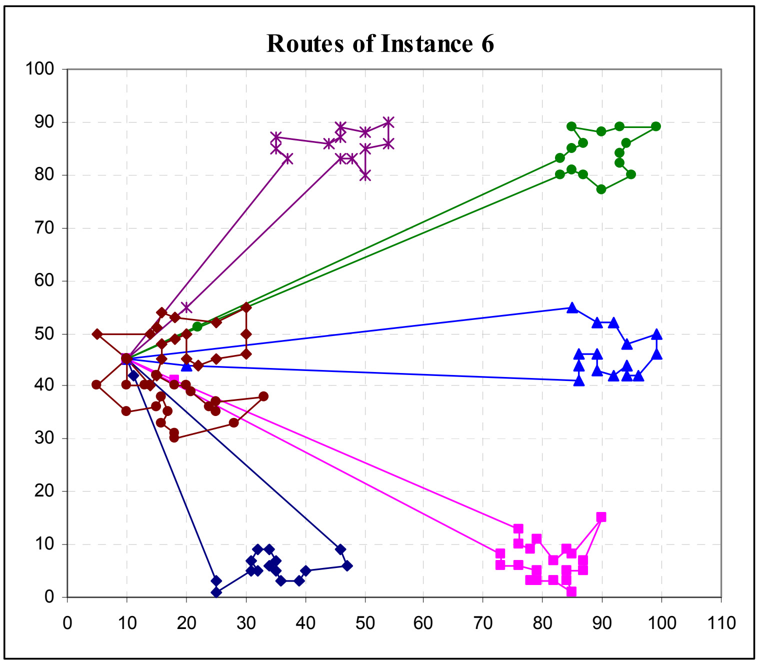
Table E5.1: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
24	2	191,85	327	0-95-3-181-73-18-146-135-92-148-163-110-25-56-32-0 0-76-187-97-161-118-72-147-106-44-55-3-0
26	2	171,42	455	0-60-26-100-71-119-10-80-131-129-169-50-12-168-81-126-0 0-126-17-130-39-109-57-189-31-162-75-9-40-0
22	2	206,02	209	0-175-46-176-35-180-69-84-134-85-164-170-11-150-0 0-38-77-165-52-11-108-132-7-51-149-0
27	2	166,92	490	0-4-87-111-58-27-153-79-15-154-83-123-13-99-179-34-0 0-34-65-8-102-178-78-177-14-133-19-128-70-167-0
23	2	181,09	200	0-120-171-47-155-36-122-166-138-48-30-54-0 0-127-125-98-45-29-124-20-138-37-88-103-5-59-0
28	2	162,79	563	0-152-139-172-173-174-82-121-21-140-94-64-28-101-2-0 0-112-66-196-186-22-141-91-113-142-114-156-93-86-157-2-0
24	2	158,4	291	0-42-68-143-137-89-185-90-41-115-160-184-192-0 0-53-198-158-197-184-190-43-199-136-191-1-194-193-0
25	2	153,51	303	0-151-117-63-107-24-145-144-74-49-182-16-96-6-0 0-6-61-33-105-195-104-23-62-116-183-67-159-188-0

E6. TEST INSTANCE 6



E1.1 Subproblems of Dror and Trudeau test instance 1.

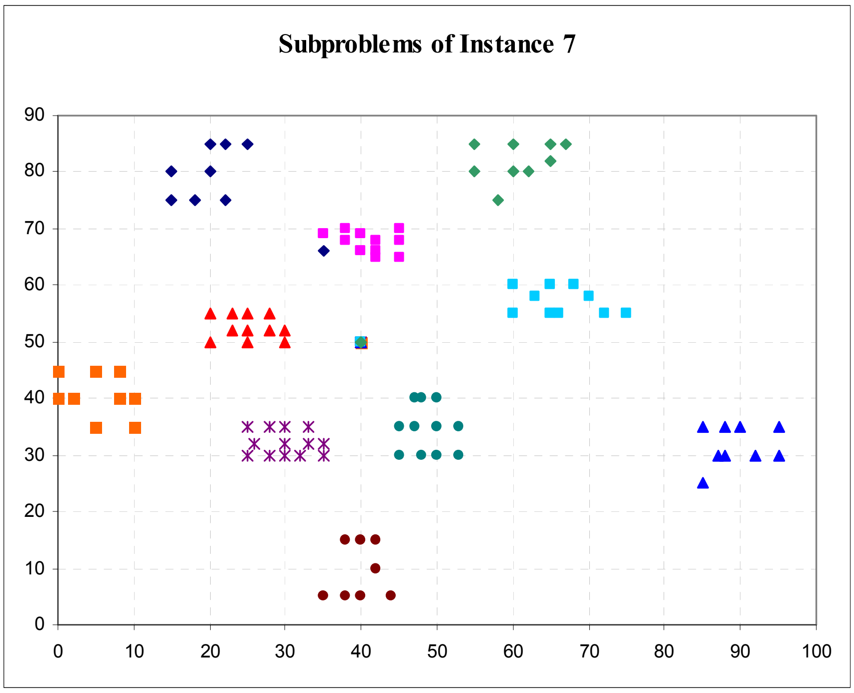


E6.2 Routes of Dror and Trudeau test instance 6.

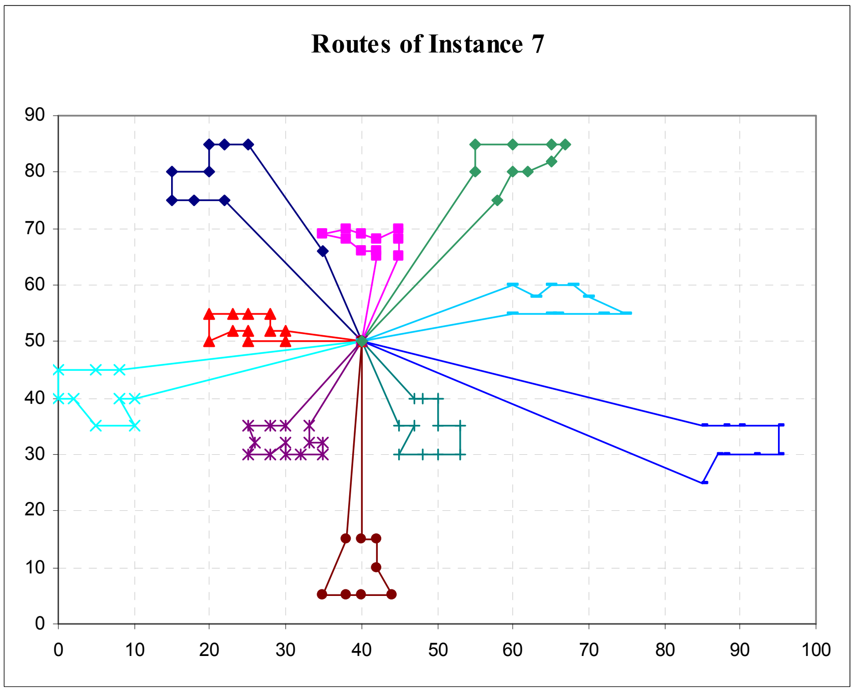
Table E6.1: Solutions of subproblems.

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
16	1	134,96	80	0-88-2-1-3-4-5-6-7-9-10-11-15-14-13-12-8-0
21	1	207,51	113	0-92-21-20-23-26-28-32-35-29-36-34-31-30-33-27-24-22-25-19-16-17-0
16	1	199,62	88	0-40-43-45-48-51-50-49-46-47-44-41-42-39-38-37-93-0
16	1	213,63	62	0-100-52-54-57-59-65-61-62-64-66-63-60-56-58-55-53-0
15	1	144,41	56	0-67-69-70-71-74-75-72-78-80-79-77-68-76-73-103-0
				0-82-111-86-87-89-91-90-18-118-114-109-108-83-113-117-84-85-112-81-119-0
36	2	161,75	229	0-120-105-106-107-104-116-98-110-115-97-94-96-99-101-102-95-87-86-0

E7. TEST INSTANCE 7



E7.1 Subproblems of Dror and Trudeau test instance 7.



E7.2 Routes of Dror and Trudeau test instance 7.

Table E7.1: Solutions of subproblems.

Number of Nodes (n)	**Number of Vehicles (v)**	**Traveling Cost**	**Computation Time**	**Route**
9	1	96,04	7	0-13-17-18-19-15-16-14-12-10-0
11	1	56,17	56	0-75-1-2-4-6-9-11-8-7-3-5-0
11	1	50,8	16	0-21-22-23-26-28-30-29-27-25-24-20-0
9	1	97,23	57	0-34-36-39-38-37-35-31-33-32-0
13	1	64,81	59	0-47-49-52-50-51-48-46-45-44-40-41-42-43-0
8	1	101,88	2	0-59-60-58-56-53-54-55-57-0
11	1	59,4	36	0-69-66-68-64-61-72-74-62-63-65-67-0
9	1	127,3	7	0-81-78-76-71-70-73-77-79-80-0
10	1	76,07	20	0-91-89-88-85-84-82-83-86-87-90-0
9	1	95,94	17	0-99-100-97-93-92-94-95-96-98-0

APPENDIX F
DETAILED SOLUTIONS OF HVRP TEST INSTANCES UNDER SPLIT DELIVERY STRATEGY

F1. TEST INSTANCE 3

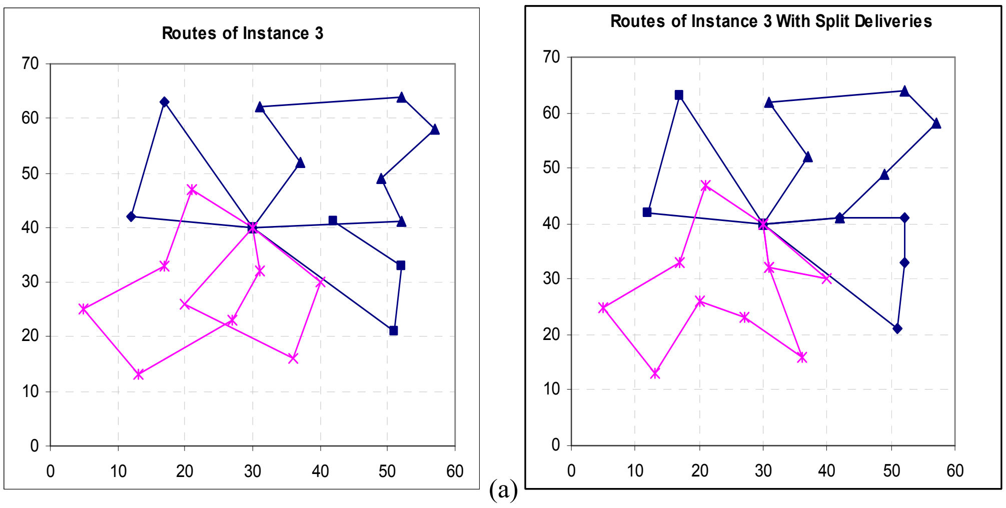
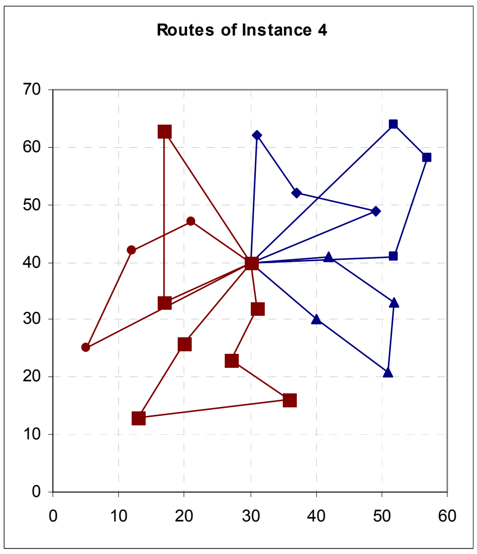


Figure F1.1: Graphical solutions of (a) non-split (b) split delivery strategy.

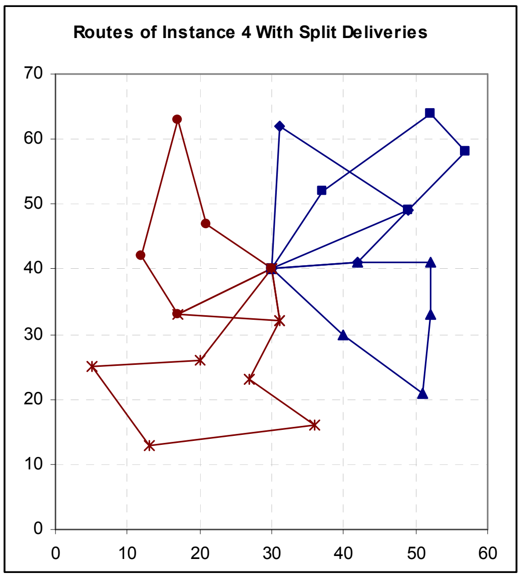
Table F1.1: Solutions of subproblems

Number of Nodes (n)	**Number of Vehicles (v)**	**Traveling Cost**	**Computation Time**	**Route**
9	2	144,84	17	0-5-12-0
				0-12-15-17-4-19-13-18-6-0
11	3	225,69	**88**	0-11-16-9-10-0
				0-14-7-0
				0-1-8-3-20-2-11-0

F2. TEST INSTANCE 4



(a)



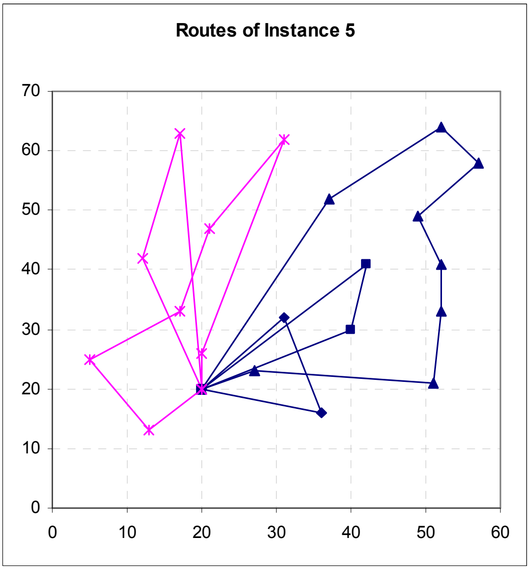
(b)

Figure F2.1: Graphical solutions of (a) non-split (b) split delivery strategy.

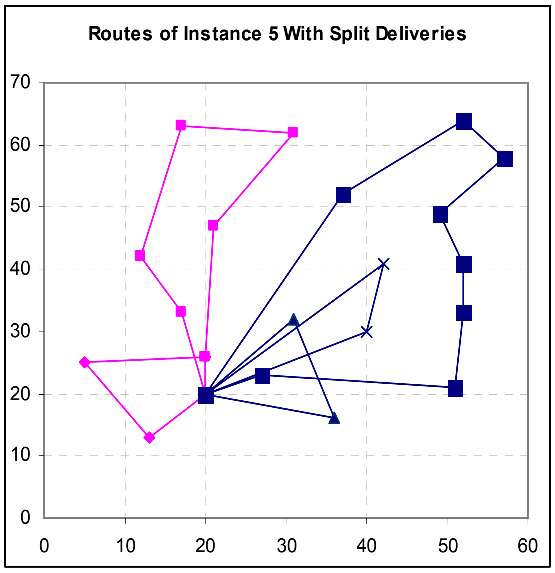
Table F2.1: Solutions of subproblems

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
10	3	211,3	**101**	0-8-2-11-0
				0-1-3-20-2-0
				0-11-16-9-10-5-0
10	3	210,6	198	0-18-12-0
				0-12-17-15-19-13-4-0
				0-18-14-7-6-0

F3. TEST INSTANCE 5



(a)



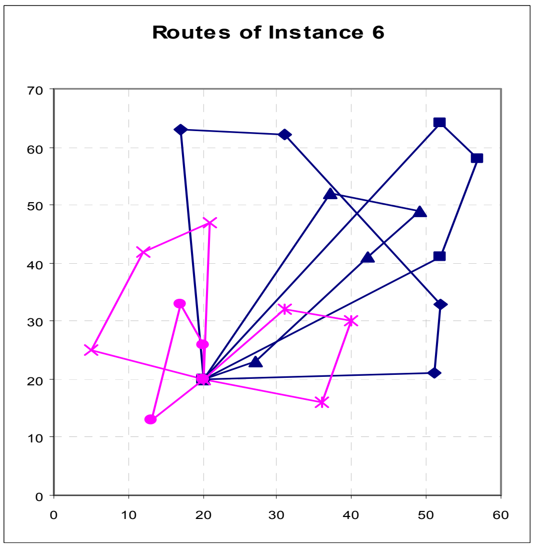
(b)

Figure F3.1: Graphical solutions of (a) non-split (b) split delivery strategy.

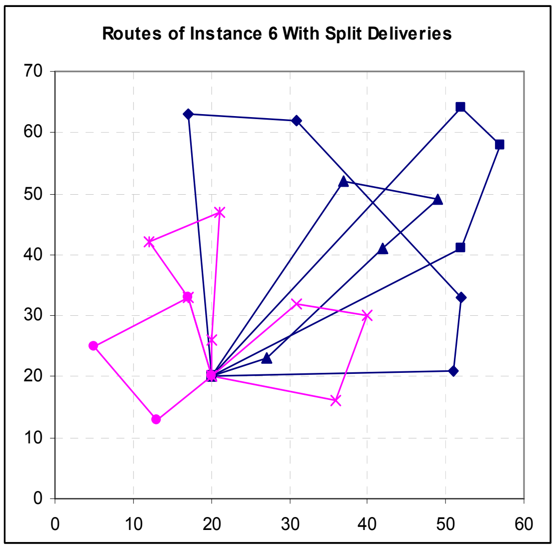
Table F3.1: Solutions of subproblems

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
8	2	149,7	10	0-19-13-4-0
				0-18-14-7-8-6-4-0
12	3	249,07	**223**	0-12-15-0
				0-5-11-0
				0-17-10-9-16-2-20-3-1-0

F4. TEST INSTANCE 6



(a)



(b)

Figure F4.1: Graphical solutions of (a) non-split (b) split delivery strategy.

Table F4.1: Solutions of subproblems

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
				0-12-5-15-0
				0-4-6-14-18-0
9	3	169,59	14	0-19-13-18-0
				0-10-9-8-7-0
				0-16-20-3-0
11	3	344,5	**281**	0-17-11-2-1-0

F5. TEST INSTANCE 13

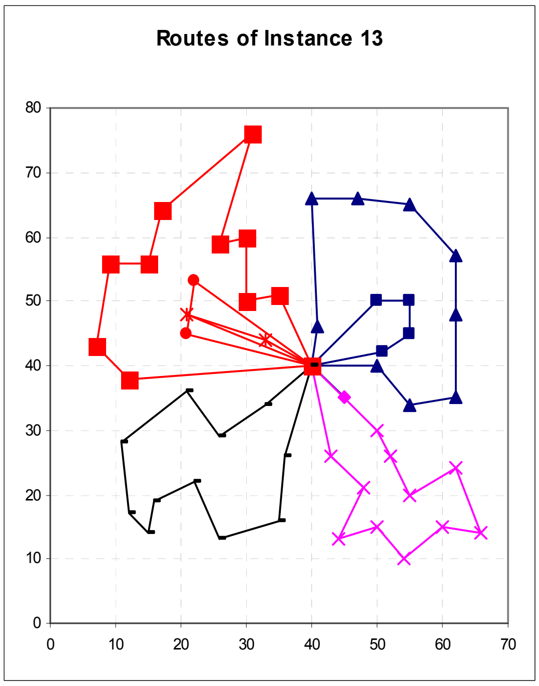


Figure F5.1: Graphical solutions of non-split delivery strategy. Split deliveries do not affect the solution.

F6. TEST INSTANCE 14

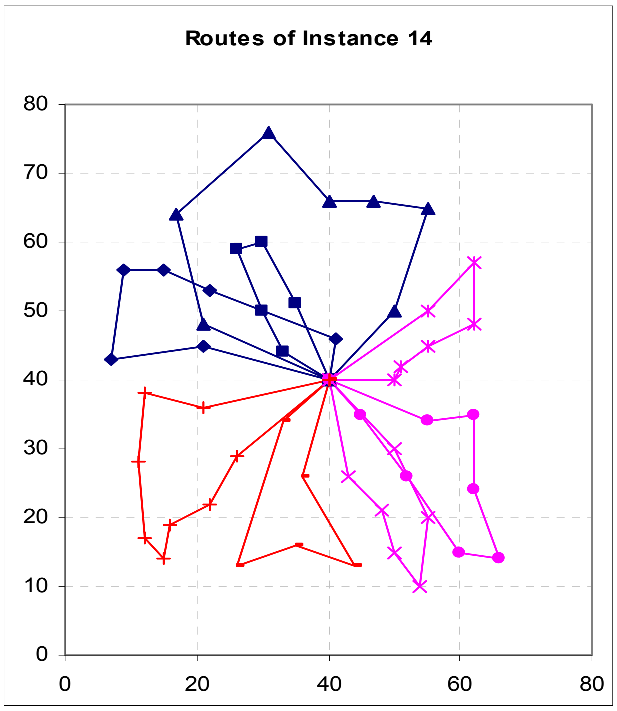
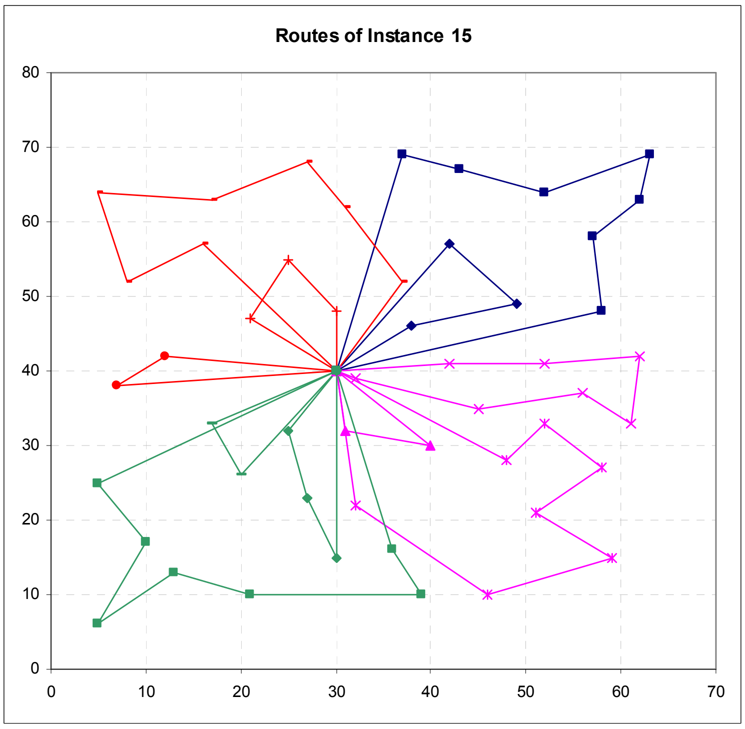
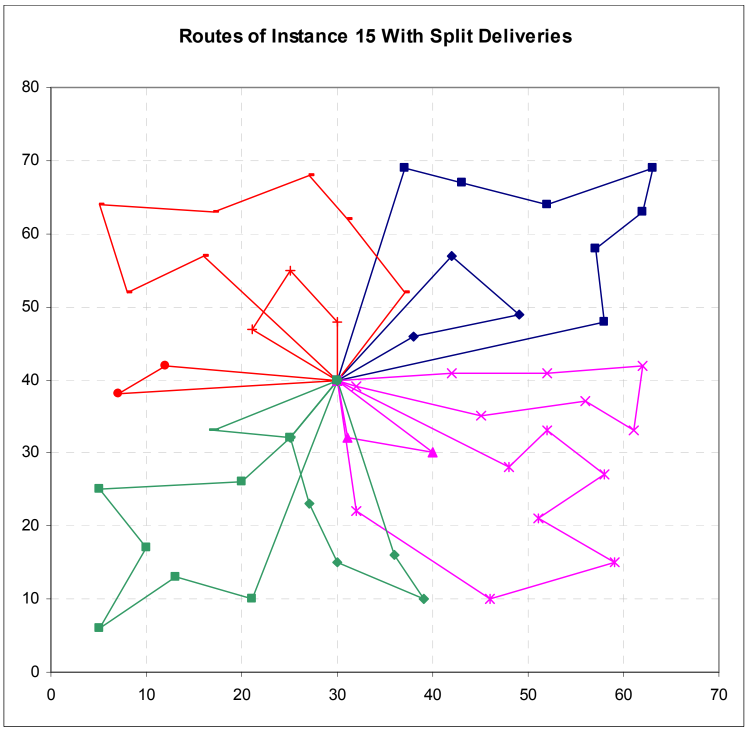


Figure F6.1: Graphical solutions of non-split delivery strategy. Split deliveries do not affect the solution.

F7. TEST INSTANCE 15



(a)



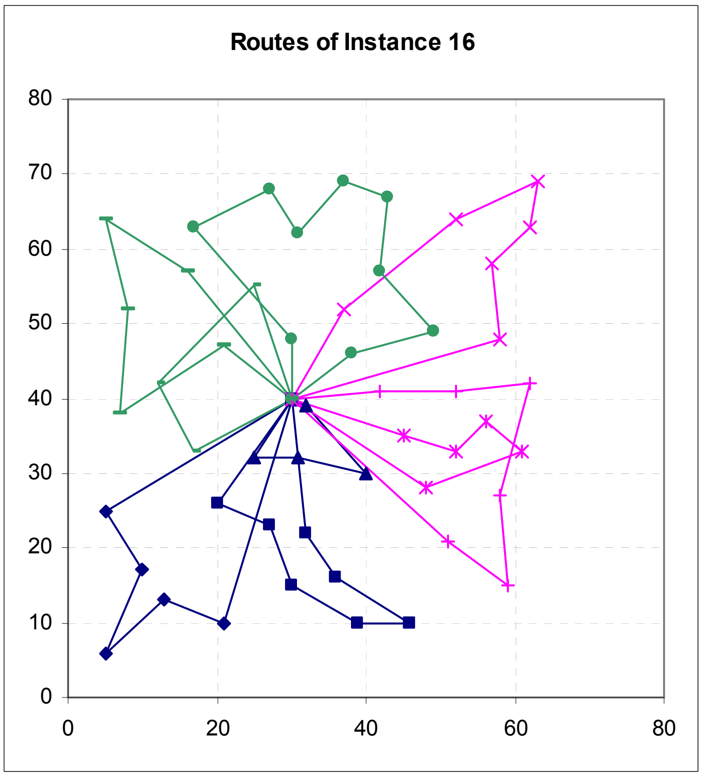
(b)

Figure F7.1: Graphical solutions of (a) non-split (b) split delivery strategy.

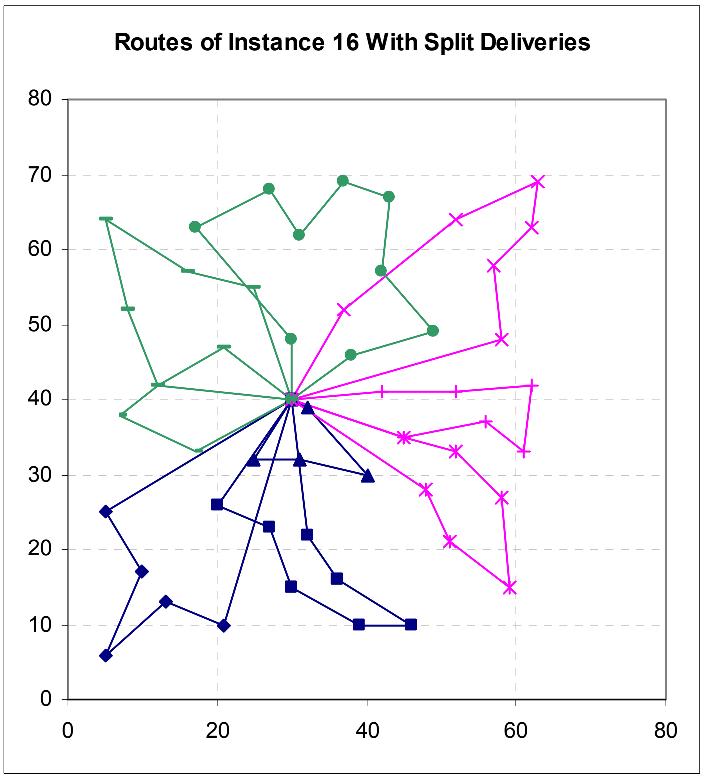
Table F7.1: Solutions of subproblems

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
				0-32-2-22-0
10	2	162,89	34	0-31-28-3-36-35-20-29-0
				0-47-17-44-45-15-0
				0-18-47-0
12	3	206,6	53	0-42-19-40-41-13-4-47-0
				0-25-14-0
				0-27-48-6-0
12	3	184,36	99	0-23-24-43-7-26-8-1-0
				0-5-12-0
				0-11-16-21-34-50-38-46-0
16	3	212,21	**247**	0-49-9-30-10-39-33-37-0

F8. TEST INSTANCE 16



(a)



(b)

Figure F8.1: Graphical solutions of (a) non-split (b) split delivery strategy.

Table F8.1: Solutions of subproblems

Number of Nodes (n)	**Number of Vehicles (v)**	**Traveling Cost**	**Computation Time**	**Route**
16	3	227,74	257	0-13-41-40-19-42-0
				0-4-17-44-45-33-15-37-0
				0-47-12-5-46-0
17	3	237,23	**409**	0-48-23-43-24-14-6-0
				0-14-25-18-0
				0-27-7-26-8-31-28-22-2-32-0
17	3	254,92	346	0-29-20-35-36-3-1-0
				0-38-9-30-39-10-49-0
				0-38-50-34-21-16-11-0

F9. TEST INSTANCE 17

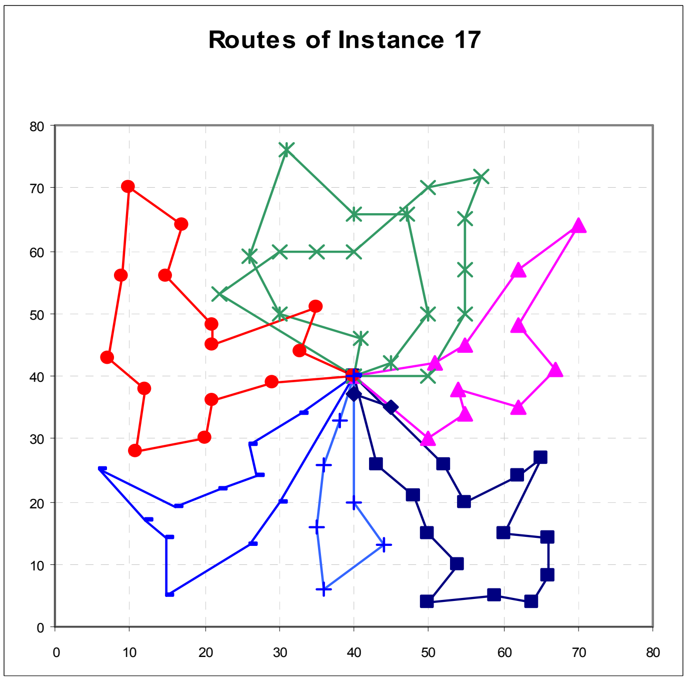


Figure F9.1: Graphical solutions of non-split delivery strategy. Split deliveries do not affect the solution.

F10. TEST INSTANCE 18

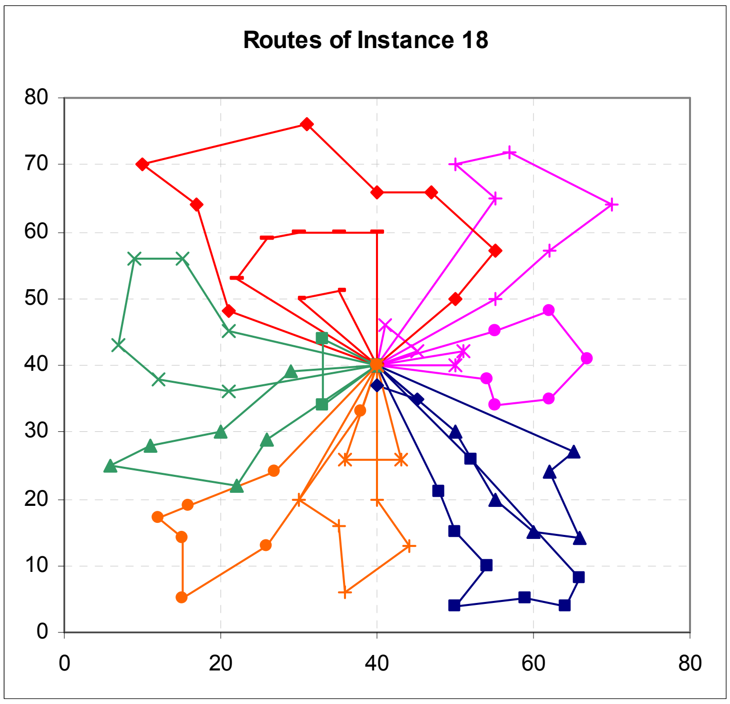
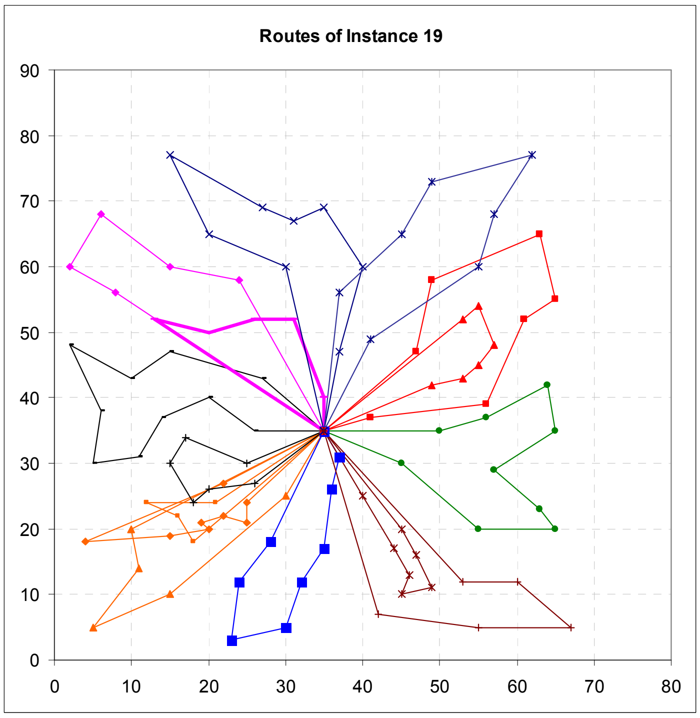
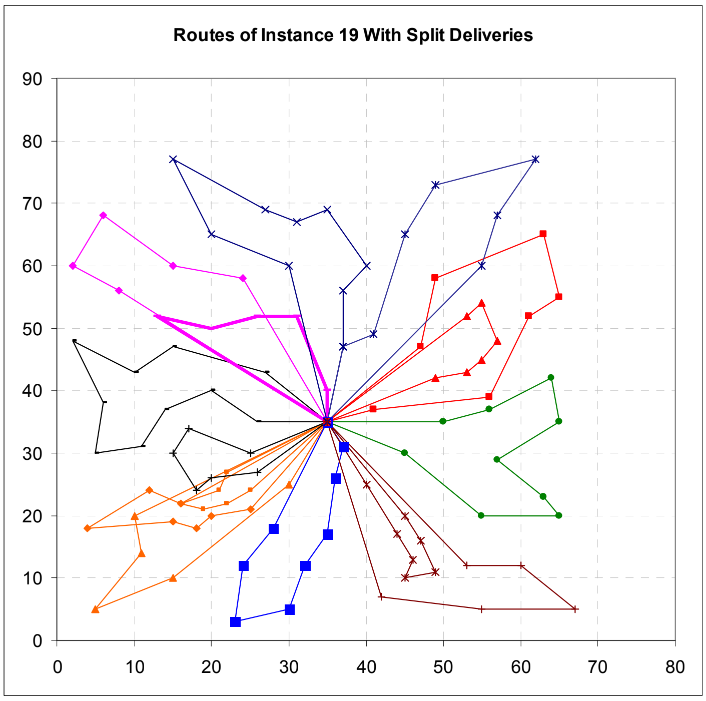


Figure F10.1: Graphical solutions of non-split delivery strategy. Split deliveries do not affect the solution.

F11. TEST INSTANCE 19



(a)



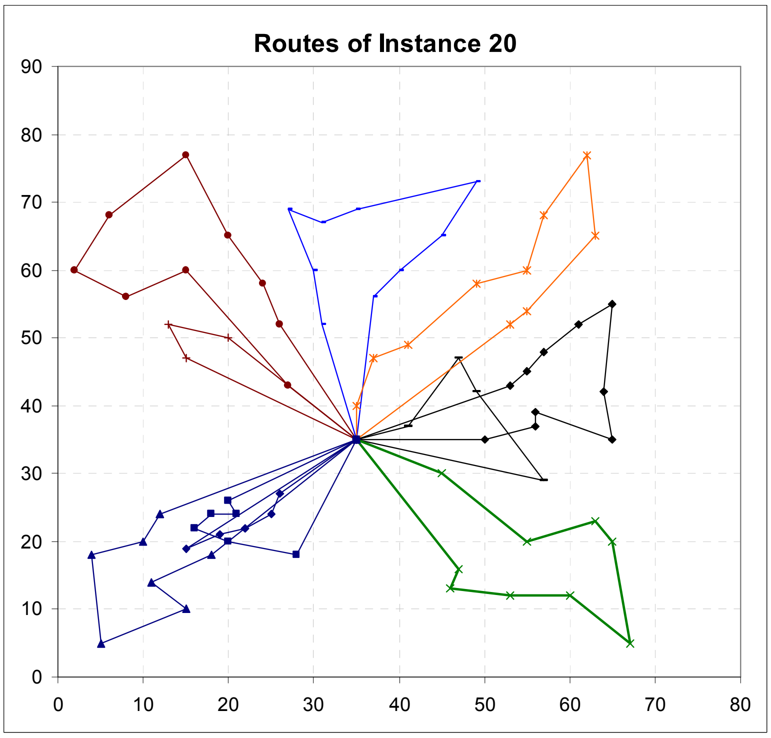
(b)

Figure F11.1: Graphical solutions of (a) non-split (b) split delivery strategy.

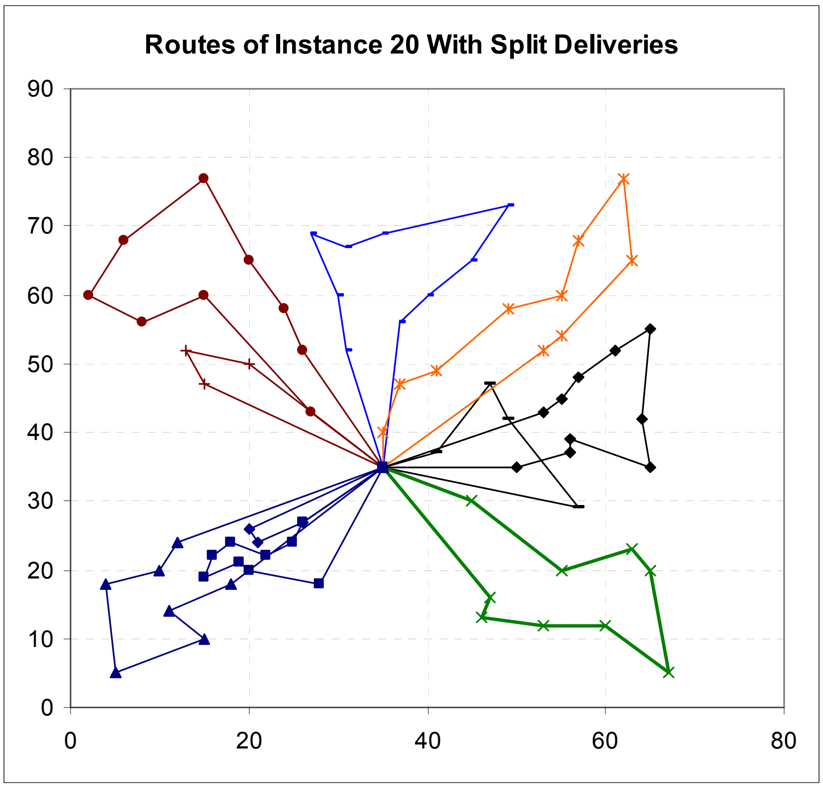
Table F11.1: Solutions of subproblems

Number of Nodes (n)	**Number of Vehicles (v)**	**Traveling Cost**	**Computation Time**	**Route**
15	2	215,74	183	0-9-71-65-66-20-1-69-0
				0-69-70-30-32-90-63-64-11-10-0
13	2	154,51	139	0-33-81-79-3-77-76-0
				0-28-68-78-34-35-51-50-0
10	2	161,17	76	0-27-31-88-7-48-0
				0-47-36-49-19-62-0
16	2	147,2	165	0-6-60-5-93-99-94-0
				0-52-82-8-46-45-17-84-83-18-89-0
9	1	95,24	65	0-26-4-25-55-54-24-29-80-12-0
12	2	159,89	112	0-41-23-67-39-56-0
				0-21-72-75-22-74-73-40-0
8	1	73,67	30	0-53-58-2-57-15-43-42-87-0
17	3	226,56	566	0-96-86-91-37-98-92-97-95-0
				0-100-85-61-59-0
				0-13-14-38-44-16-0

F12. TEST INSTANCE 20



(a)



(b)

Figure F12.1: Graphical solutions of (a) non-split (b) split delivery strategy.

Table F12.1: Solutions of subproblems

Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
16	2	102,2	201	0-13-57-2-40-53-0 0-97-42-43-15-41-22-23-75-73-21-58-0
14	2	160,29	153	0-54-76-50-28-0 0-77-3-79-78-34-29-24-68-80-12-0
12	2	151,51	89	0-52-19-47-36-49-64-11-62-88-0 0-7-48-82-0
12	2	106,9	67	0-6-96-84-17-45-46-8-83-0 0-5-60-18-89-0
9	1	100,6	5	0-26-4-55-25-67-39-56-74-72-0
9	1	172,89	11	0-31-10-63-90-32-66-20-30-70-0
10	1	138,39	23	0-27-69-1-51-9-71-65-35-81-33-0
18	3	206,99	261	0-94-59-99-0 0-94-95-92-93-85-91-98-37-87-0 0-100-44-14-38-86-16-61-0

APPENDIX G
OPL STUDIO CODES OF THRESHOLD ALGORITIHM FOR VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

G1. CODE OF MODEL TWM

```
setting searchStrategy = SBS;
SheetConnection sheet("C:\C102.xls");
int Demandpoints = 12;
int Noofvehicles=1;
int Ready=1236;
range Stores 1..Demandpoints;
range Total 0..Demandpoints;
range Vehicles 1..Noofvehicles;
float NewDistance[Total, Total] from SheetRead(sheet,"distancesub1");
int Latest[Stores] from SheetRead(sheet,"latest");
 int Earliest[Stores] from SheetRead(sheet,"earliest");
 int Service[Total] from SheetRead(sheet,"service");
 var int X[Total,Total] in 0..Noofvehicles;
var int  T[Total] in 0..1236;
var int W[Total] in 0..0;
solve
{
//Each city is visited exactly once;
forall (i in Stores) sum (j in Total) (X[i,j]>0)=1;
forall (j in Stores) sum (i in Total) (X[i,j]>0)=1;

//Each vehicle visits the depot exactly once;
forall (v in Vehicles) sum (i in Total) (X[i,0]=v) = 1;
forall (v in Vehicles) sum (j in Total) (X[0,j]=v) = 1;

//A vehicle leaves the city that it enters;
forall (i in Total) (sum (j in Total) (X[i,j]) - sum (j in Total) (X[j,i]))=0;
//Capacity of vehicles should not be exceeded;forall(v in Vehicles)
 (sum(i in Total) Demand[i]*(sum(j in Total) (X[i,j]=v))) <=Capacityofvehicle[v];
T[0]=0;
W[0]=0;

//If going from node i to node j is infeasible, then corresponding variable should be
0;
forall (i in Total, j in Total) X[i,j]<=50*M[i,j];

//Arrival time to a node must be within its speciefied time window;
forall(i in Stores) (T[i]+W[i])<=Latest[i];
forall(i in Stores) (T[i]+W[i])>=Earliest[i];

//Arrival time to a node must be greater than or equal to (arrival time of the previous
node+waiting time+service time+traveling time);
forall(j in Stores) (sum (v in Vehicles, i in Total)
   ((X[i,j]=v)*(T[i]+W[i]+Service[i]+NewDistance[i,j])))<=T[j];

//Arrival to the depot must be greater than or equal to (arrival time of the previous
node+waiting time+service time+traveling time);
forall(v in Vehicles) (sum(j in Stores)
   ((X[j,0]=v)*(T[j]+W[j]+Service[j]+NewDistance[j,0])))<=Ready;
};
```

APPENDIX H
OPL STUDIO CODES OF SETCOVERING ALGORITIHM FOR VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

H1. CODE OF MODEL SETCOVERING

```
int Demandpoints = 25;
int Noofvehicles=10;
range Stores 1..Demandpoints;
range Total 0..Demandpoints;
import int noofovercapacity;
import int exceedingdemand[0..noofovercapacity,0..Demandpoints];
import int noofovertime;
import int exceedingtime[0..noofovercapacity,0..Demandpoints];
import int rhs[0..noofovertime];
import int noofouts;
import int outoftimewindow[0..noofovercapacity,0..Demandpoints];
import int rhs2[0..noofouts];
import int nosubtour;
import int rhs3[0..nosubtour];
import int subtours[0..nosubtour,Stores];
import int M[Total,Total] ;
import float NewDistance[Total, Total] ;
import int Demand[Total] ;
range subtourrange 0..nosubtour;
range outrange 0..noofouts;
range overtimerange 0..noofovertime;
range overcaprange 0..noofovercapacity;
var int X[Total,Total] in 0..1;
minimize sum(i in Total, j in Total) ((X[i,j])*NewDistance[i,j])
subject to
{//All nodes should be visited exactly once;
 forall (i in Stores) sum (j in Total) X[i,j]=1;
 forall (j in Stores) sum (i in Total) X[i,j]=1;
 //Number of vehicles should not be exceeded;
sum(j in Total) X[0,j]<=Noofvehicles;
sum(j in Total) X[0,j]=sum(i in Total) X[i,0];

//A vehicle can go from a node only to one of its time window satisfactory neighbours;
 forall (i in Total, j in Total) X[i,j]<=M[i,j];

//Elimination of overcapacities;
 forall(t in overcaprange) sum(i in 0..Demandpoints-1)
(X[exceedingdemand[t,i],exceedingdemand[t,i+1]]*Demand[exceedingdemand[t,i]])<=1000;
 forall(t in overcaprange) sum(i in 0..Demandpoints-1)
(X[exceedingdemand[t,i+1],exceedingdemand[t,i]]*Demand[exceedingdemand[t,i]])<=1000;

//Elimination of overtimes;
 forall(t in overtimerange) sum(i in 0..Demandpoints-1)
        (X[exceedingtime[t,i],exceedingtime[t,i+1]])<=rhs[t];
forall(t in overtimerange) sum(i in 0..Demandpoints-1)
        (X[exceedingtime[t,i+1],exceedingtime[t,i]])<=rhs[t];


//Elimination of time window violations;
 forall(t in outrange) (sum(i in 0..Demandpoints-1)
        (X[outoftimewindow[t,i],outoftimewindow[t,i+1]])<=rhs2[t]);
 forall(t in outrange) (sum(i in 0..Demandpoints-1)
        (X[outoftimewindow[t,i+1],outoftimewindow[t,i]])<=rhs2[t]);

//Elimination of subtours;
 forall(t in subtourrange)  sum(i in 1..Demandpoints-1)
        (X[subtours[t,i],subtours[t,i+1]])<=rhs3[t];
 forall(t in subtourrange)  sum(i in 1..Demandpoints-1)
        (X[subtours[t,i+1],subtours[t,i]])<=rhs3[t];};
```

H2. CODE SCRIPT ALGORITM FOR MODEL TWM

```
SheetConnection sheet("C:\C204.xls");
int n:=25;
int vehicles:=10;
range Stores 1..n;
range Total 0..n;
int M[Total,Total] from SheetRead(sheet,"nereye2");
float NewDistance[Total, Total] from SheetRead(sheet,"newdistance2");
int Demand[Total] from SheetRead(sheet,"demandtotal2");
int Latest[Stores] from SheetRead(sheet,"latest2");
int Earliest[Stores] from SheetRead(sheet,"earliest2");
int Service[Total] from SheetRead(sheet,"service2");
int due:=230;
int capa:=200;
int j:=0;
int i:=0;
int k:=0;
int v:=0;
int l:=1;
int p:=0;
int z:=0;
int z1:=0;
int z2:=0;
int nj:=0;
int deger:=0;
int deger2:=0;
int solution:=0;
int solution2:=0;
int solution3:=0;
int toplamdemand:=0;
float totaltime:=0;
int noofovercapacity:=-1;
int noofovertime:=-1;
int noofouts:=-1;
int matris[1..vehicles, 0..n];
float arrive[1..n];
float leave[1..n];
int count:=0;
ofile pinar("pinar.txt") ;
int say:=1;
int saygenel:=1;
int rhsgecici:=0;
int subtour:=0;
int generalsubtour:=0;
int nosubtour:=-1;
int matrisilk[Stores];
int matrisgenel[Stores];
k:=0;
j:=0;
i:=1;
Open int exceedingdemand[0..noofovercapacity,0..n];
Open int exceedingtime[0..noofovertime,0..n];
Open int outoftimewindow[0..noofouts,0..n];
Open int subtours[0..nosubtour,Stores];
Open int rhs[0..noofovertime];
Open int rhs2[0..noofouts];
Open int rhs3[0..nosubtour];
//Model m("twdecomposition.mod") editMode;
repeat
{
repeat{
 Model m("twdecomposition.mod");
 m.solve();
```

```
forall(m in Stores)
 matrisgenel[m]:=0;
saygenel:=1;
i:=1;
generalsubtour:=0;
cout << m.objectiveValue() << endl;
pinar << m.objectiveValue() << endl;

 while i<=n do
 {count:=0;
  forall(m in 1..n)
   if i=matrisgenel[m] then {count:=1; break;}
  if count=0 then
           {k:=i; say:=1; subtour:=0;
            j:=0;
            while j<=n do
            {if m.X[k,j]=1 then
                          {matrisilk[say]:=k;
                          matrisgenel[saygenel]:=k;
                          saygenel:=saygenel+1;
                          k:=j;
                          if j=0 then {say:=1; k:=i;
                                   break;}
                          if k=matrisilk[1] then {rhsgecici:=say-1;
                                        say:=say+1;
                                        while say<=n do
                                         {matrisilk[say]:=k;
                                          say:=say+1;}
                                        subtour:=1; break;}
                          forall(m in Stores)
                            if k=matrisgenel[m] then {count:=1; break;}
                          if count=1 then break;
```

```
                        j:=-1;
                        say:=say+1;}
   j:=j+1;}
   if subtour=1 then {nosubtour:=nosubtour+1;
                   subtours.addh();
                   rhs3.addh();
                   rhs3[nosubtour]:=rhsgecici;
                   forall(t in Stores)
                    subtours[nosubtour,t]:=matrisilk[t];
                   generalsubtour:=1;}}
                   i:=i+1;}

if generalsubtour=0 then {forall(i in Total)
                   {forall(j in Total)
                     cout << m.X[i,j] << "   " ;
                    cout << endl;}
                    cout << m.objectiveValue();
                    break;}
}until 0;
cout << "full solution" << endl;
j:=0;
i:=0;
k:=0;
Model m("twdecomposition.mod") ;
forall(i in 1..vehicles)
 forall(j in 0..n)
  matris[i,j]:=0;
i:=0;
j:=0;
v:=0;
k:=0;
l:=1;
```

```
p:=0;
while m.solve() do
{
  while v<vehicles do
  {while j<=n do
   {if m.X[i,j]=1 then
              {cout << i << "-" ;
               pinar << i << "-" ;
               matris[l,p]:=i;
               p:=p+1;
               if i=0 then k:=j;
               i:=j;
               if j=0 then
                    break;
               j:=-1;}
   j:=j+1;}
   j:=k+1; i:=0; v:=v+1; l:=l+1; p:=0;
   cout << endl;
   pinar << endl;
   cout << v << endl;
   pinar << v << endl;}
break;
 }
cout << "Objective value=" << m.objectiveValue() << endl;
solution:=noofovercapacity;
forall(i in 1..vehicles)
 {forall(j in 0..n)
  {z:=matris[i,j];
  toplamdemand:=toplamdemand+Demand[z];}
  if toplamdemand>capa then
                 {noofovercapacity:=noofovercapacity+1;
                  exceedingdemand.addh();
```

```
                        cout << "deneme" << endl;
                        forall(nj in 0..n)
                        exceedingdemand[noofovercapacity,nj]:=matris[i,nj];}
   toplamdemand:=0;}
solution2:=noofovertime;
forall(i in 1..vehicles)
 {forall(j in 0..n-1)
  {z1:=matris[i,j];
   z2:=matris[i,j+1];
   totaltime:=totaltime+NewDistance[z1,z2];
   if z2=0 then break;
   if Earliest[z2]>totaltime then
                           totaltime:=Earliest[z2];
   arrive[z2]:=totaltime;
   totaltime:=totaltime+Service[z2];
   leave[z2]:=totaltime; }
  if totaltime>due then
             {noofovertime:=noofovertime+1;
              exceedingtime.addh();
              rhs.addh();
              forall(nj in 0..n)
              exceedingtime[noofovertime,nj]:=matris[i,nj];
              nj:=1;
              deger:=0;
              repeat{
                    deger:=deger+1;
                    nj:=nj+1;
                    if nj=100 then {deger:=deger+1; break;}}
               until (matris[i,nj]=0);
             rhs[noofovertime]:=deger;}
   totaltime:=0;}
forall(j in 1..n)
```

```
 {cout<< "Arrival to " << j << " = " << arrive[j] << "   Leave from " << j << " = " <<
leave[j] << endl;
  pinar<< "Arrival to " << j << " = " << arrive[j] << "   Leave from " << j << " = " <<
leave[j] << endl;}
solution3:=noofouts;
forall(j in 1..n)
 {if arrive[j]>Latest[j] then
                        {forall(i in 1..vehicles)
                          forall(k in 0..n)
                           if matris[i,k]=j then
                                          {noofouts:=noofouts+1;
                                           outoftimewindow.addh();
                                           rhs2.addh();
                                           forall(nj in 0..n)
                                           outoftimewindow[noofouts,nj]:=matris[i,nj];
                                           nj:=1;
                                           deger2:=0;
                                           repeat{
                                                 deger2:=deger2+1;
                                                 nj:=nj+1;
                                                 if nj=100 then {deger2:=deger2+1; break;}}
                                           until matris[i,nj]=0;
                                           rhs2[noofouts]:=deger2;
                                           break;}
                        }}
cout << endl;
if noofovercapacity=solution then
                         if solution2=noofovertime then
                                                 if noofouts=solution3 then {cout << endl << "true
                                                                            solution" << endl;
                                                                         break;}
}until 0;
```

APPENDIX I

DETAILED SOLUTIONS TO SOLOMON TEST INSTANCES

I1. Subproblems of C100 Series

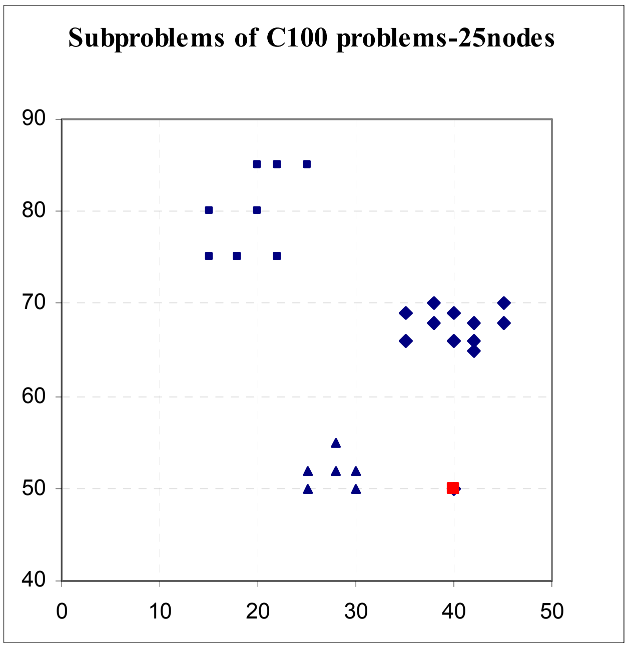


Figure I1.1 Subproblems of C100-25 nodes

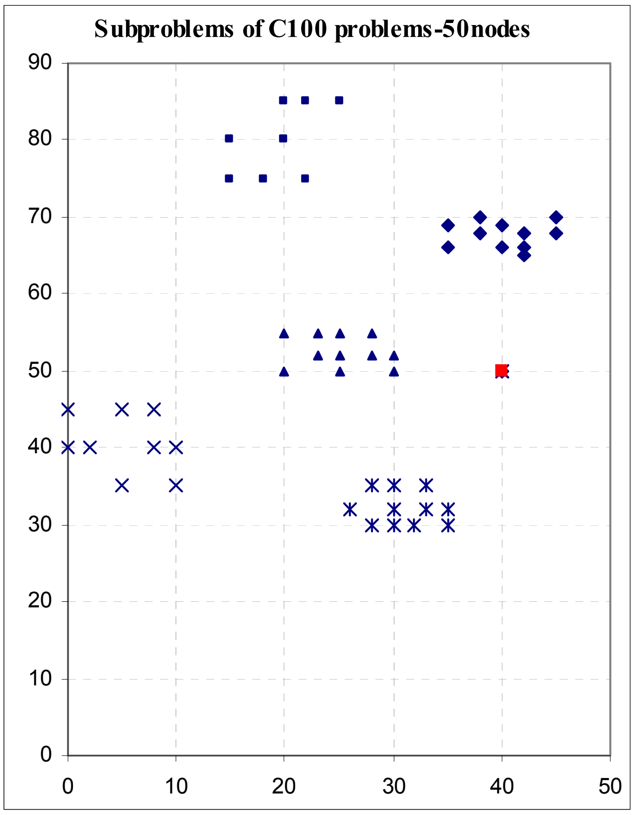


Figure I1.2 Subproblems of C100-50 nodes

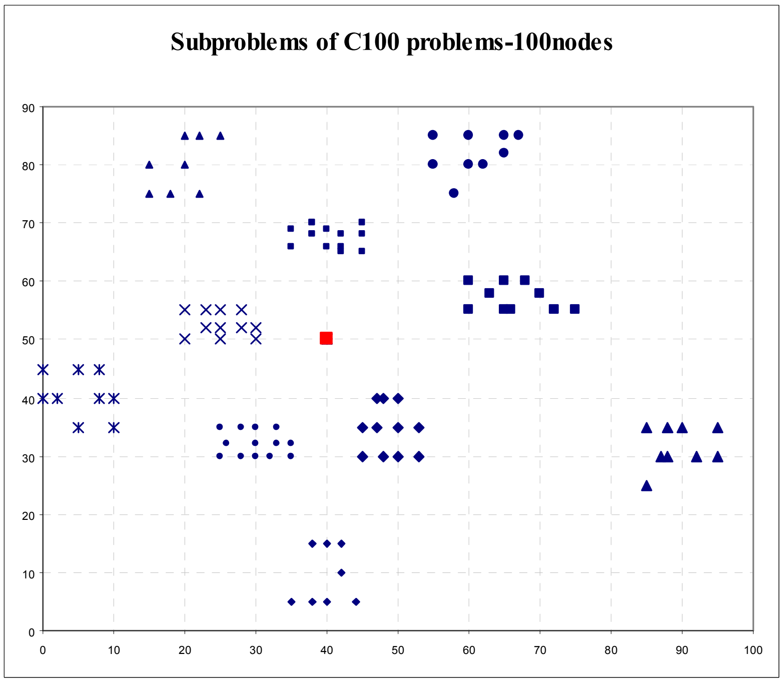


Figure I1.3 Subproblems of C100-100 nodes

.

I2. C101, C105, C106, C107, C108, C109

Solutions of C101, C105, C106, C107, C108 and C109 are same.

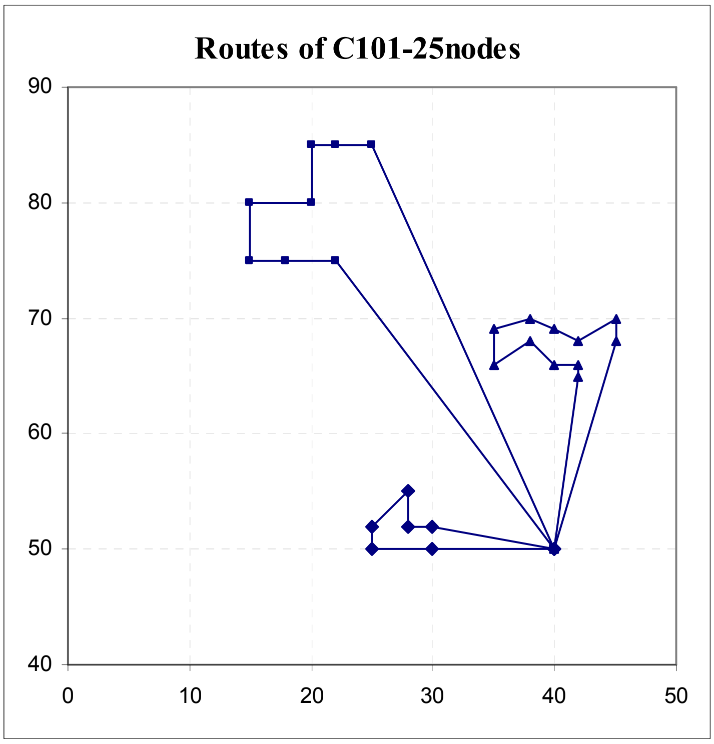


Figure I2.1: Routes of 25 node problems

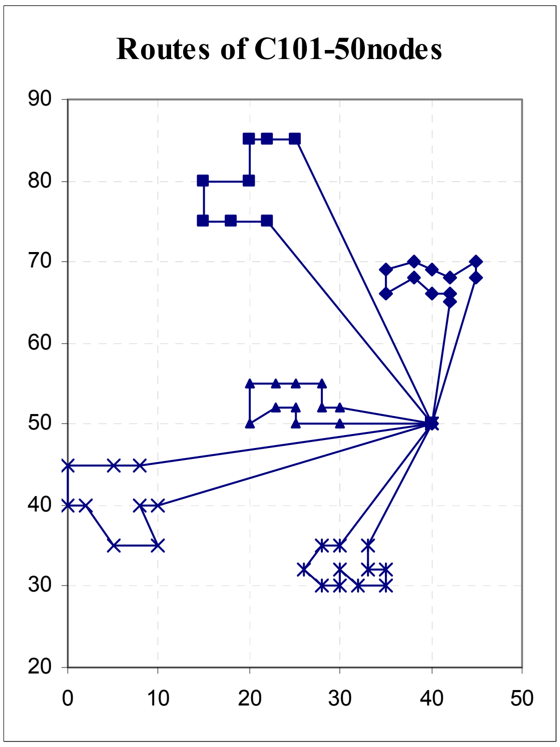


Figure I2.2: Routes of 50 node problems

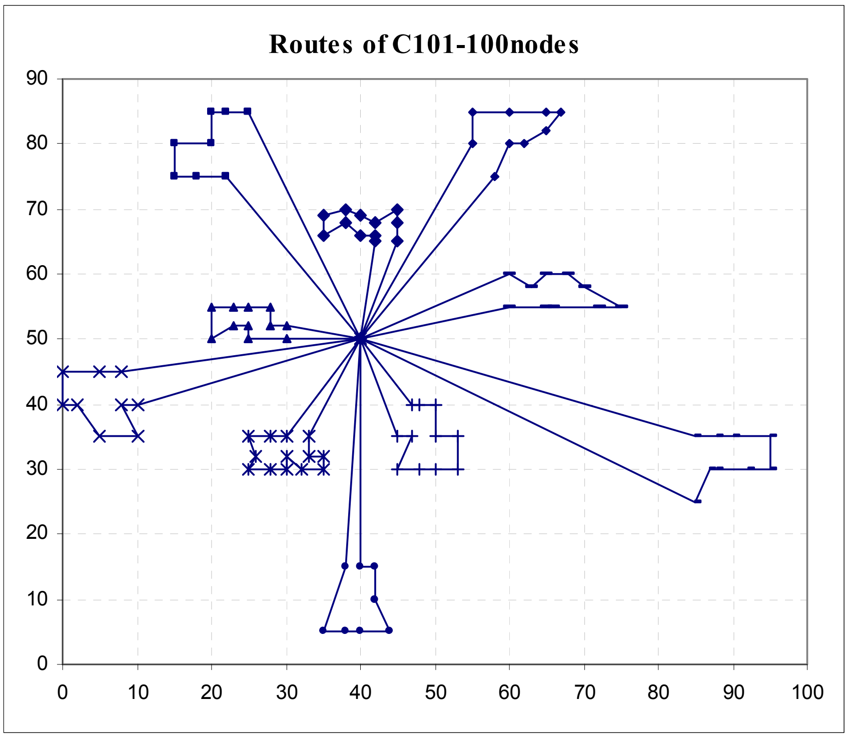


Figure I2.3: Routes of 100 node problems

Table I2.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
C101-25	11	1	59,49	<=1	0-5-3-7-8-10-11-9-6-4-2-1-0
	8	1	95,81	<=1	0-13-17-18-19-15-16-14-12-0
	6	1	36,44	<=1	0-20-24-25-23-22-21-0
C101-50	11	1	59,49	<=1	0-5-3-7-8-10-11-9-6-4-2-1-0
	8	1	95,81	<=1	0-13-17-18-19-15-16-14-12-0
	11	1	50,80	<=1	0-20-24-25-27-29-30-28-26-23-22-21-0
	9	1	97,23	<=1	0-32-33-31-35-37-38-39-36-34-0
	11	1	59,84	<=1	0-43-42-41-40-44-46-45-48-50-49-47-0
C101-100	12	1	59,62	<=1	0-5-3-7-8-10-11-9-6-4-2-1-75-0
	8	1	95,81	<=1	0-13-17-18-19-15-16-14-12-0
	11	1	50,80	<=1	0-20-24-25-27-29-30-28-26-23-22-21-0
	9	1	97,23	<=1	0-32-33-31-35-37-38-39-36-34-0
	13	1	64,81	<=1	0-43-42-41-40-44-46-45-48-51-50-52-49-47-0
	8	1	101,88	<=1	0-57-55-54-53-56-58-60-59-0
	11	1	59,4	<=1	0-67-65-63-62-74-72-61-64-68-66-69-0
	9	1	127,3	<=1	0-81-78-76-71-70-73-77-79-80-0
	10	1	76,07	<=1	0-90-87-86-83-82-84-85-88-89-91-0
	9	1	95,94	<=1	0-98-96-95-94-92-93-97-100-99-0

I3. C102, C103, C104

Solutions of C102, C103 and C104 are same.

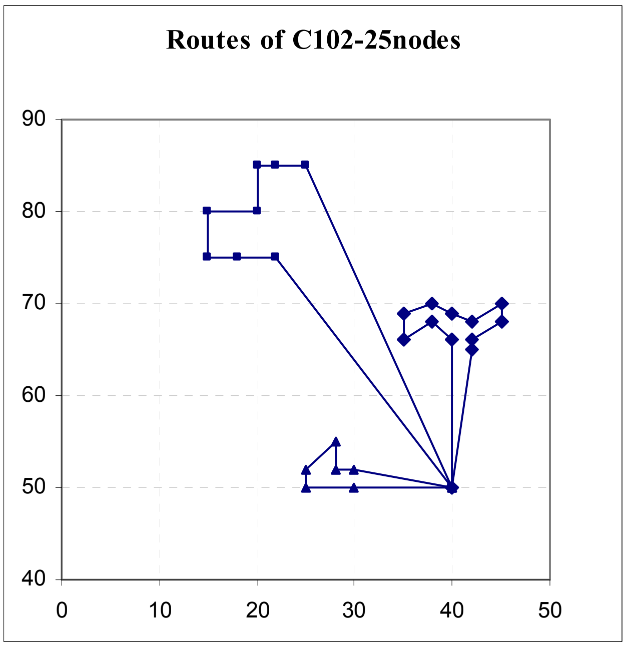


Figure I3.1: Routes of 25 node problems

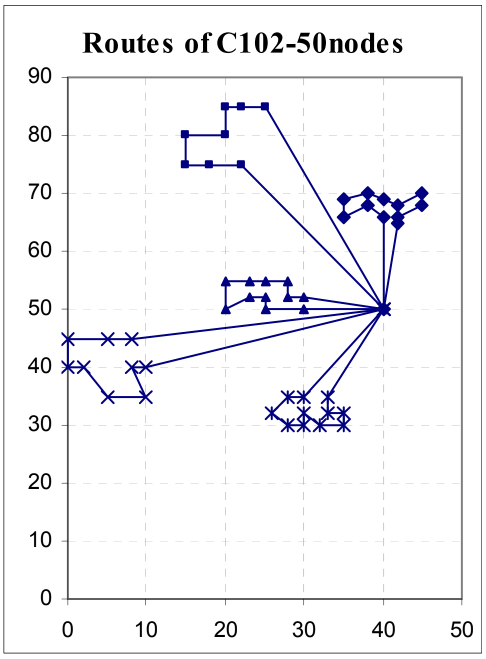


Figure I3.2: Routes of 50 node problems

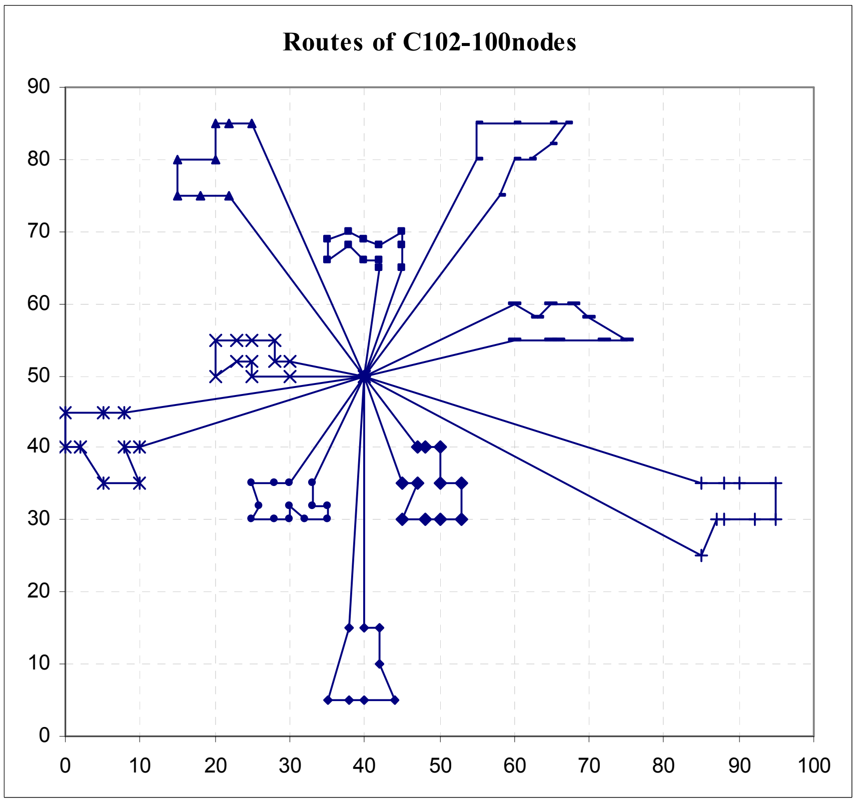


Figure I3.3: Routes of 100 node problems

Table I3.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
C102-25	11	1	58,41	<=1	0-7-8-10-11-9-6-4-2-1-3-5-0
	8	1	95,81	<=1	0-13-17-18-19-15-16-14-12-0
	6	1	36,44	<=1	0-20-24-25-23-22-21-0
C102-50	11	1	58,41	<=1	0-7-8-10-11-9-6-4-2-1-3-5-0
	8	1	95,81	<=1	0-13-17-18-19-15-16-14-12-0
	11	1	50,80	<=1	0-20-24-25-27-29-30-28-26-23-22-21-0
	9	1	97,23	<=1	0-32-33-31-35-37-38-39-36-34-0
	11	1	59,84	<=1	0-43-42-41-40-44-46-45-48-50-49-47-0
C102-100	12	1	59,62	<=1	0-5-3-7-8-10-11-9-6-4-2-1-75-0
	8	1	95,81	<=1	0-13-17-18-19-15-16-14-12-0
	11	1	50,80	<=1	0-20-24-25-27-29-30-28-26-23-22-21-0
	9	1	97,23	<=1	0-32-33-31-35-37-38-39-36-34-0
	13	1	64,81	4	0-43-42-41-40-44-46-45-48-51-50-52-49-47-0
	8	1	101,88	<=1	0-57-55-54-53-56-58-60-59-0
	11	1	59,4	<=1	0-67-65-63-62-74-72-61-64-68-66-69-0
	9	1	127,3	<=1	0-81-78-76-71-70-73-77-79-80-0
	10	1	76,07	<=1	0-90-87-86-83-82-84-85-88-89-91-0
	9	1	95,94	<=1	0-98-96-95-94-92-93-97-100-99-0

I4. Subproblems of C200 Series

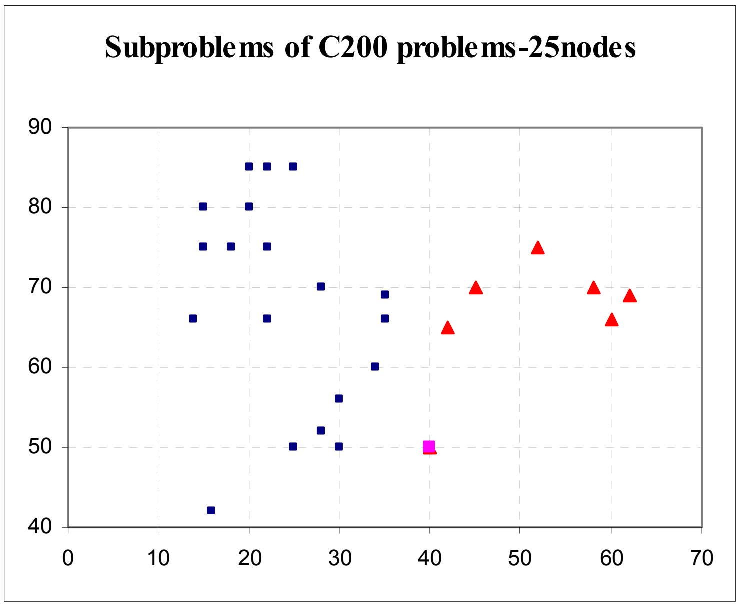


Figure I4.1 Subproblems of C200-25 nodes

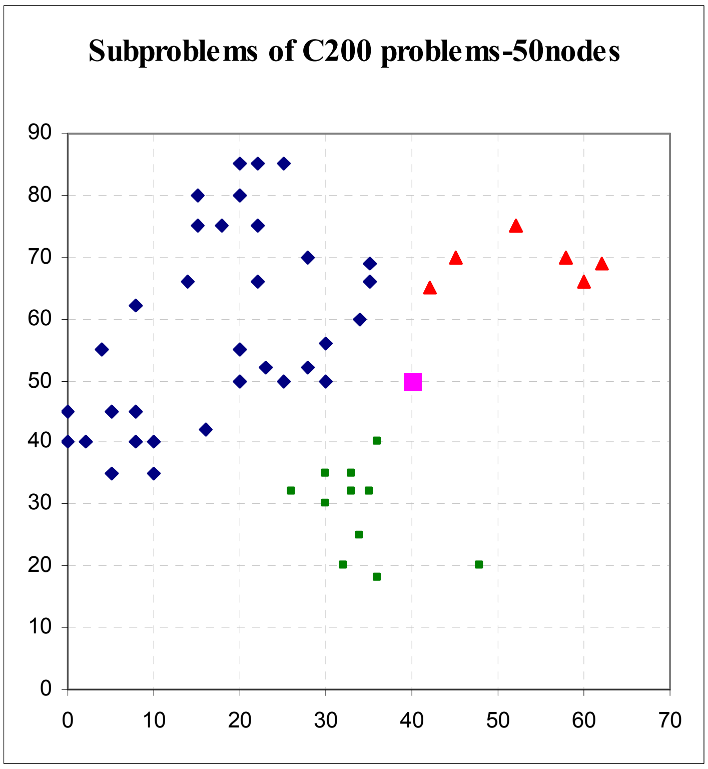


Figure I4.2 Subproblems of C200- 50 nodes

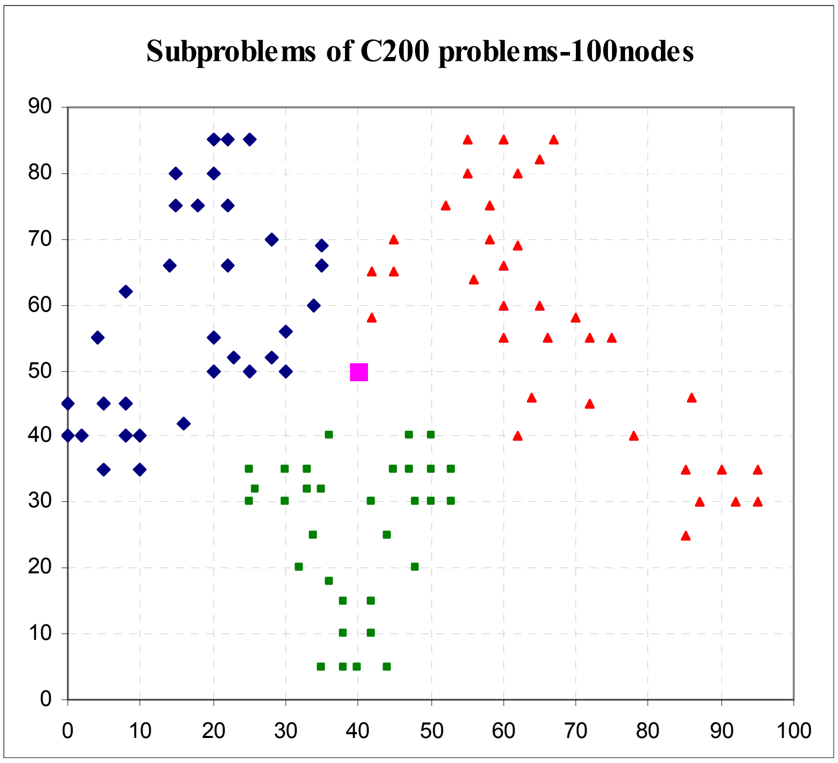


Figure I4.3 Subproblems of C200- 100 nodes

I5. C201, C202, C204, C205, C206, C207

Solutions of C201, C202, C204, C205, C206 and C207 are same.

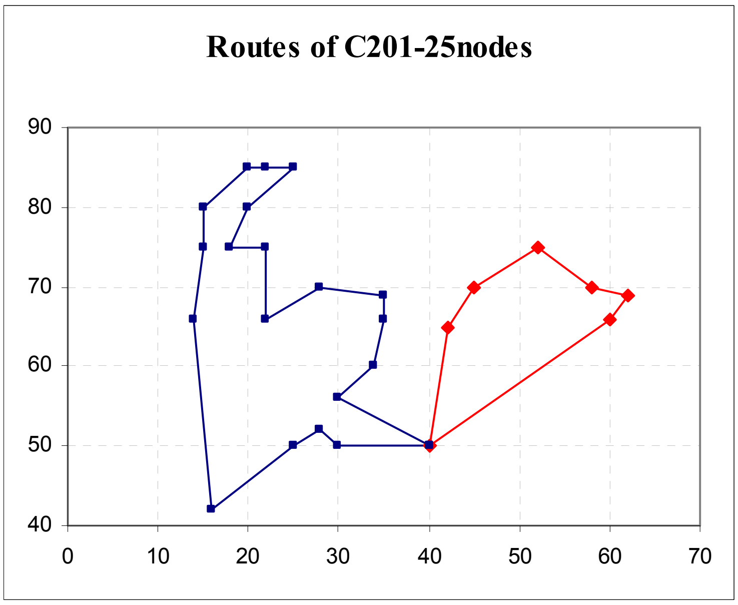


Figure I5.1: Routes of 25 node problems

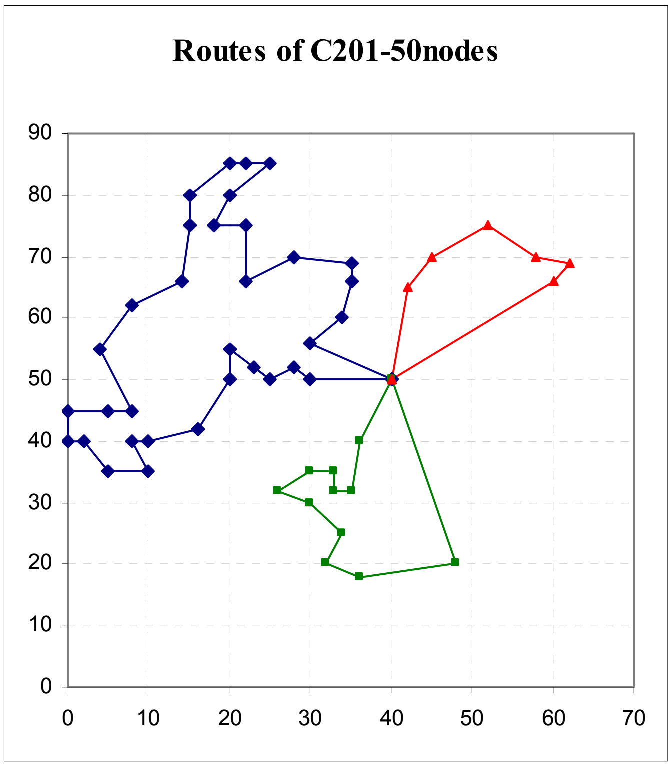


Figure I5.2: Routes of 50 node problems

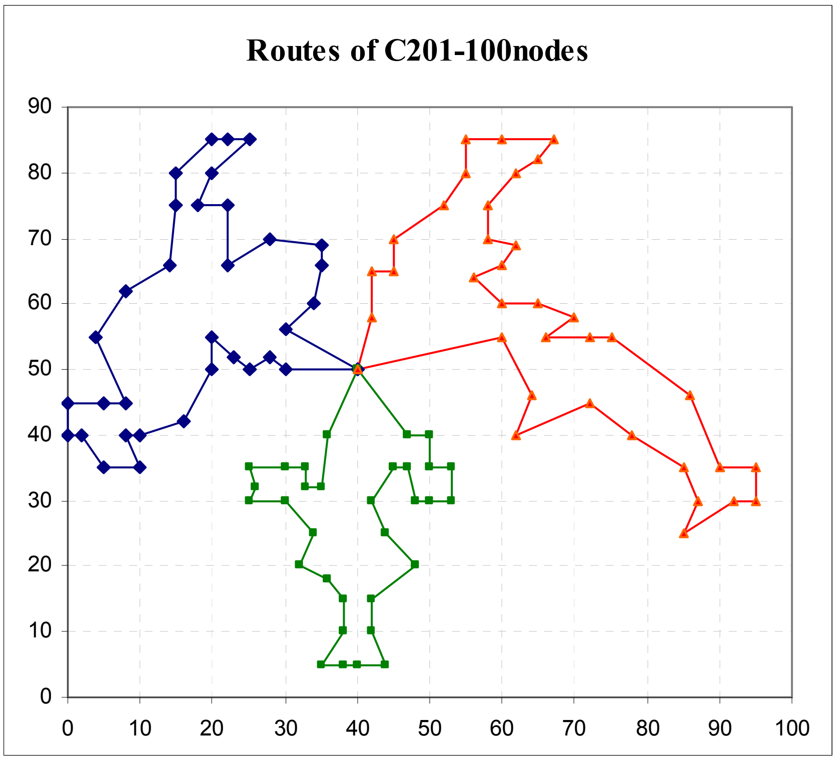


Figure I5.3: Routes of 100 node problems

Table I5.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
C201-25	6	1	70,72	<=1	0-5-2-1-7-3-4-0
	19	1	144,83	<=1	0-20-22-24-6-23-18-19-16-14-12-15-17-13-25-9-11-10-8-21-0
C201-50	6	1	70,72	<=1	0-5-2-1-7-3-4-0
	11	1	95,78	<=1	0-49-40-44-46-45-50-47-43-42-41-48-0
	33	1	195,30	118	0-20-22-24-27-30-29-6-32-33-31-35-37-38-39-36-34-28-26-23-18-19-16-14-12-15-17-13-25-9-11-10-8-21-0
C201-100	33	1	195,30	118	0-20-22-24-27-30-29-6-32-33-31-35-37-38-39-36-34-28-26-23-18-19-16-14-12-15-17-13-25-9-11-10-8-21-0
	32	1	158,05	126	0-67-63-62-74-72-61-64-66-69-68-65-49-55-54-53-56-58-60-59-57-40-44-46-45-51-50-52-47-43-42-41-48-0
	35	1	238,21	281	0-93-5-75-2-1-99-100-97-92-94-95-98-7-3-4-89-91-88-84-86-83-82-85-76-71-70-73-80-79-81-78-77-96-87-90-0

I6. C203, C208

Solutions of C203 and C208 are same.

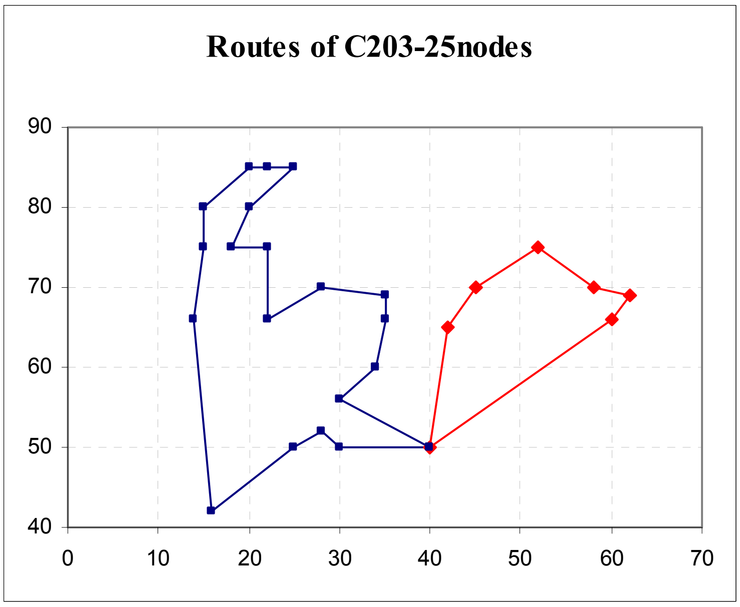


Figure I6.1: Routes of 25 node problems

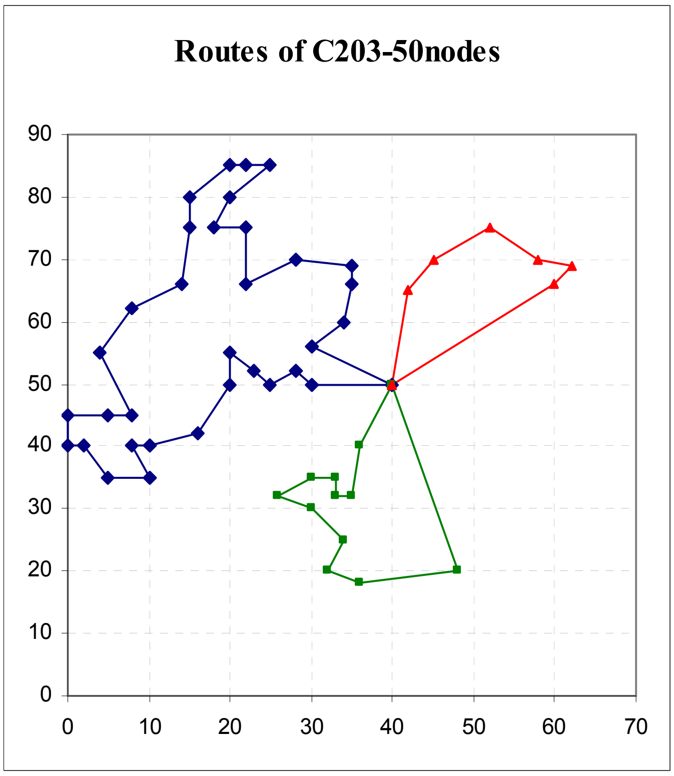


Figure I6.2: Routes of 50 node problems

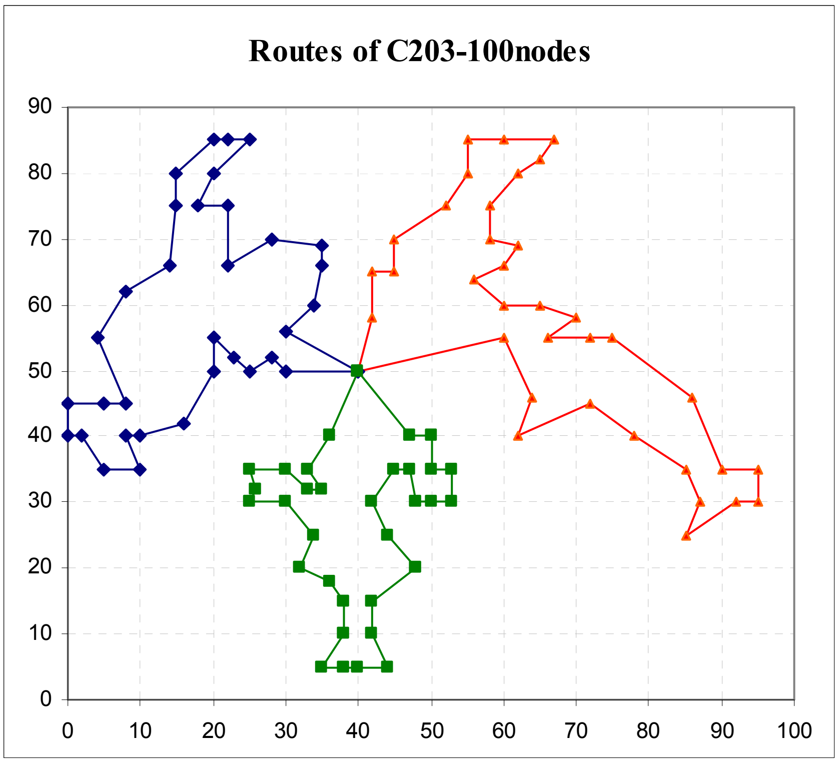


Figure I6.3: Routes of 100 node problems

Table I6.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Computation Time	Route
C203-25	6	1	70,72	<=1	0-5-2-1-7-3-4-0
	19	1	144,83	21	0-20-22-24-6-23-18-19-16-14-12-15-17-13-25-9-11-10-8-21-0
C203-50	6	1	70,72	<=1	0-5-2-1-7-3-4-0
	11	1	95,78	<=1	0-49-40-44-46-45-50-47-43-42-41-48-0
	33	1	195,30	206	0-20-22-24-27-30-29-6-32-33-31-35-37-38-39-36-34-28-26-23-18-19-16-14-12-15-17-13-25-9-11-10-8-21-0
C203-100	33	1	195,30	206	0-20-22-24-27-30-29-6-32-33-31-35-37-38-39-36-34-28-26-23-18-19-16-14-12-15-17-13-25-9-11-10-8-21-0
	32	1	157,66	188	0-67-63-62-74-72-61-64-66-69-68-65-49-55-54-53-56-58-60-59-57-40-44-46-45-51-50-52-47-42-41-43-48-0
	35	1	238,21	415	0-93-5-75-2-1-99-100-97-92-94-95-98-7-3-4-89-91-88-84-86-83-82-85-76-71-70-73-80-79-81-78-77-96-87-90-0

I7. Nodes of R100 Series

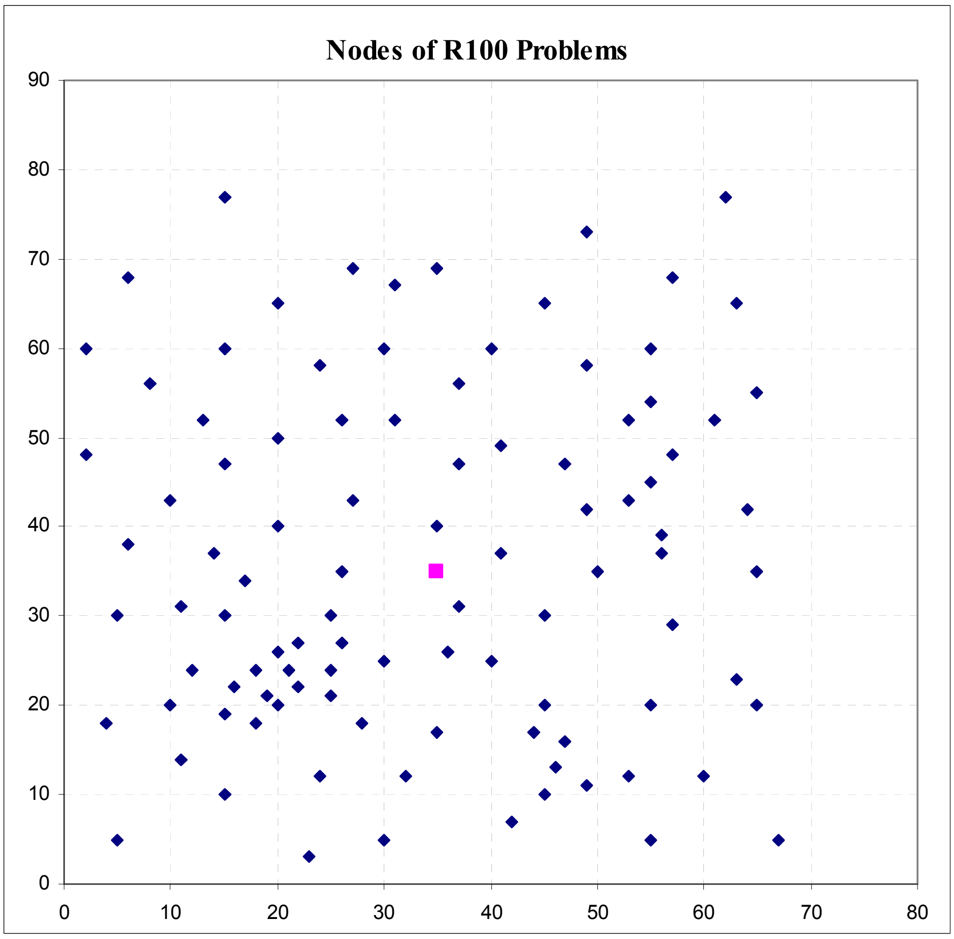


Figure I7.1: Nodes of R 100 problems

I8. R101

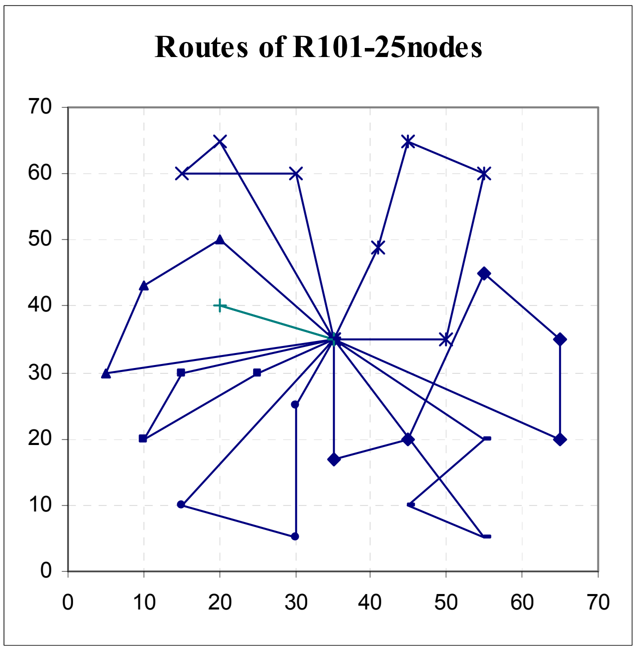


Figure I8.1: Routes of R101-25 nodes

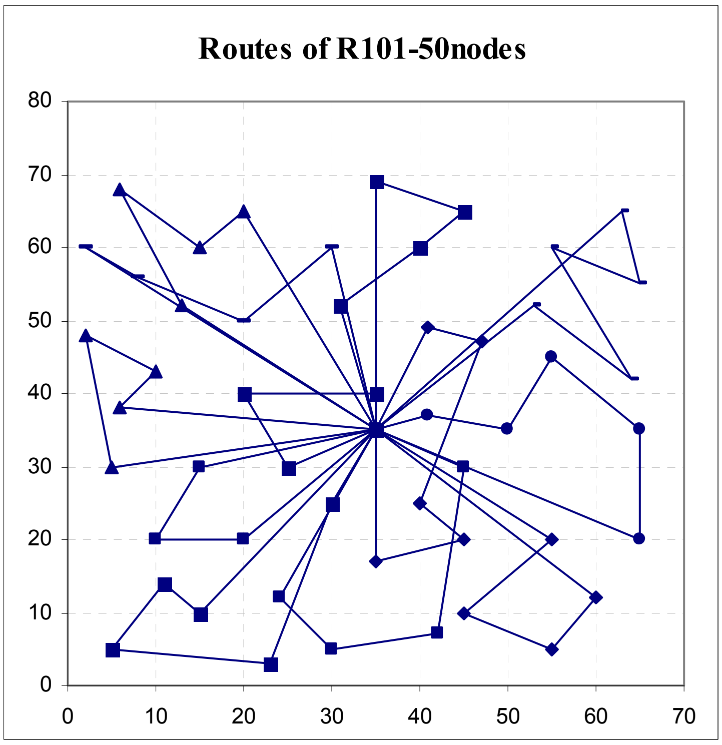


Figure I8.2: Routes of R101-50 nodes

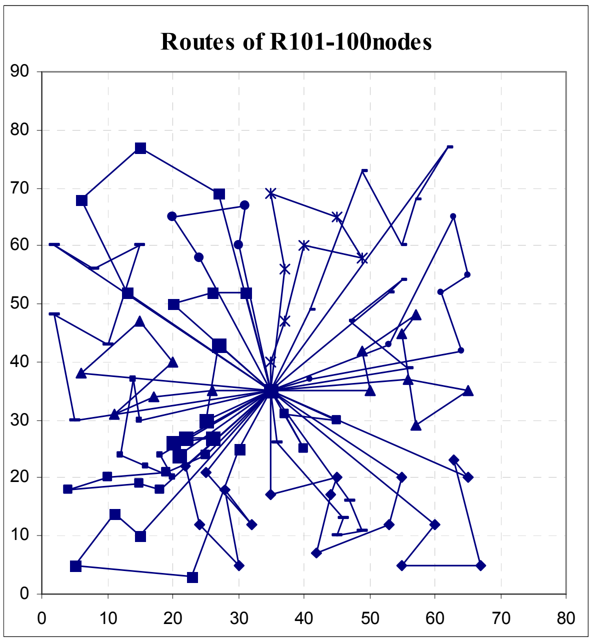


Figure I8.3: Routes of R101-100 nodes

Table I8.1: Solutions of subproblems

Problem	Traveling Cost	Route	Problem	Traveling Cost	Route
R101-25	618,33	0-2-21-3-24-25-0	R101-100	1642,88	0-2-21-73-41-56-4-0
		0-5-16-6-0			0-5-83-61-85-37-93-0
		0-7-8-17-0			0-12-76-79-3-54-24-80-0
		0-11-19-10-0			0-14-44-38-43-13-0
		0-12-9-20-1-0			0-27-69-30-51-20-32-70-0
		0-14-15-13-0			0-28-29-78-34-35-77-0
		0-18-0			0-31-88-7-0
		0-23-22-4-0			0-33-81-50-68-0
R101-50	1046,70	0-2-21-40-50-1-0			0-36-47-19-8-46-17-0
		0-5-16-37-0			0-39-23-67-55-25-0
		0-11-19-49-48-0			0-40-53-26-0
		0-14-44-38-43-13-0			0-45-82-18-84-60-89-0
		0-27-18-6-0			0-52-6-0
		0-28-12-3-24-25-0			0-59-99-94-96-0
		0-31-30-20-32-0			0-62-11-90-10-0
		0-33-29-9-34-35-0			0-63-64-49-48-0
		0-36-47-7-10-0			0-65-71-9-66-1-0
		0-39-23-22-4-0			0-72-75-22-74-58-0
		0-42-15-41-26-0			0-92-42-15-87-57-97-0
		0-45-8-46-17-0			0-95-98-16-86-91-100-0

I9. R105

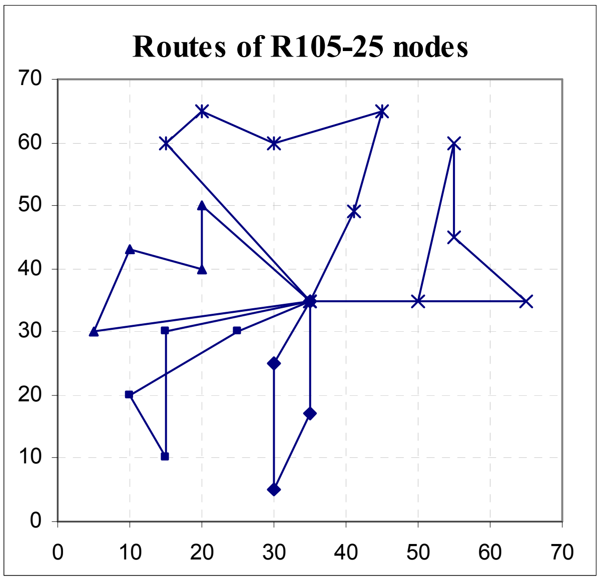


Figure I9.1: Routes of R105-25 nodes

Table I9.1: Solutions of subproblems

Problem	Traveling Cost	Route
R105-25	531,54	0-2-15-13-0
		0-5-14-16-6-0
		0-7-18-8-17-0
		0-12-9-3-24-0
		0-19-11-10-20-1-0
		0-21-23-22-4-25-0

I10. Subproblems of RC Series

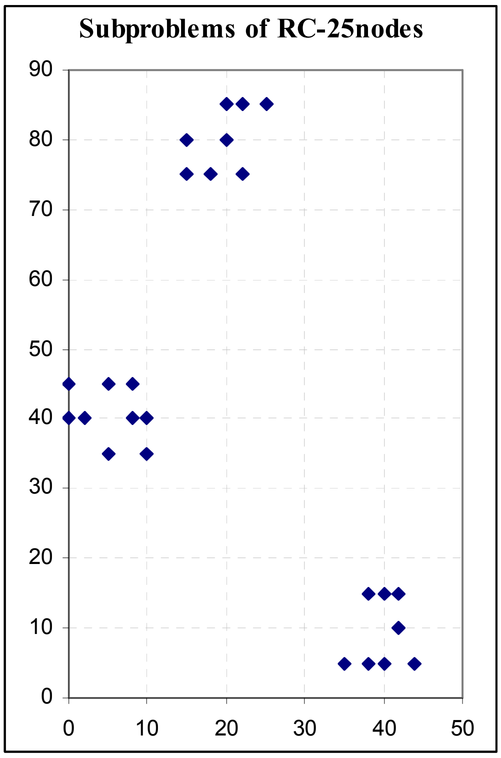


Figure I10.1: Subproblems of RC-25 nodes

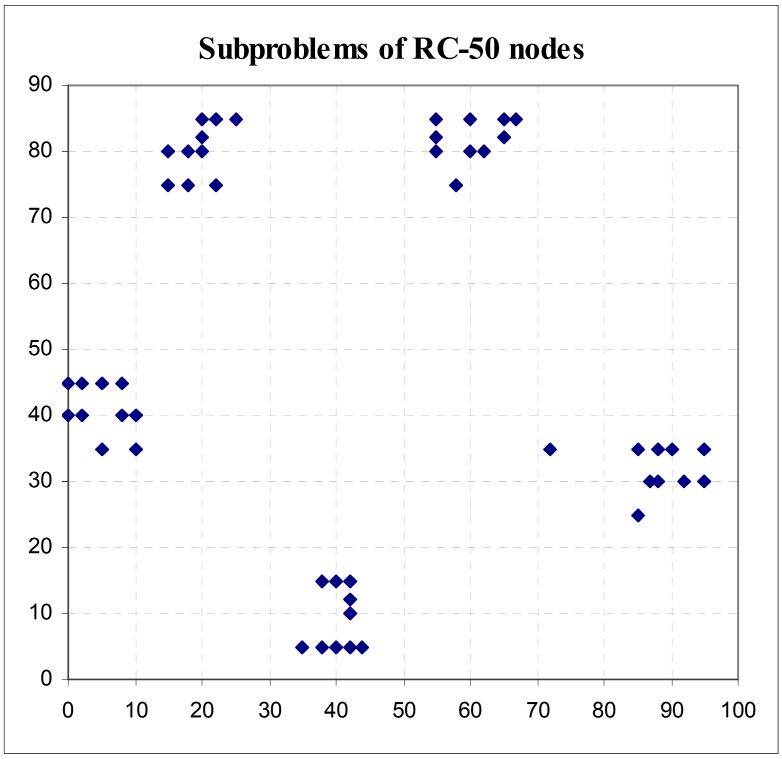


Figure I10.2: Subproblem of RC-50 nodes

I11. RC101

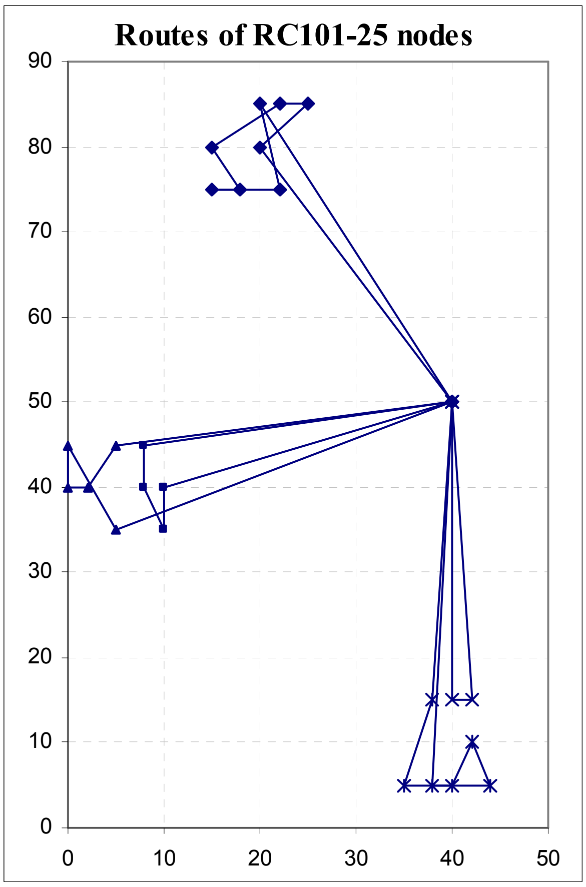


Figure I11.1: Routes of RC101-25 nodes

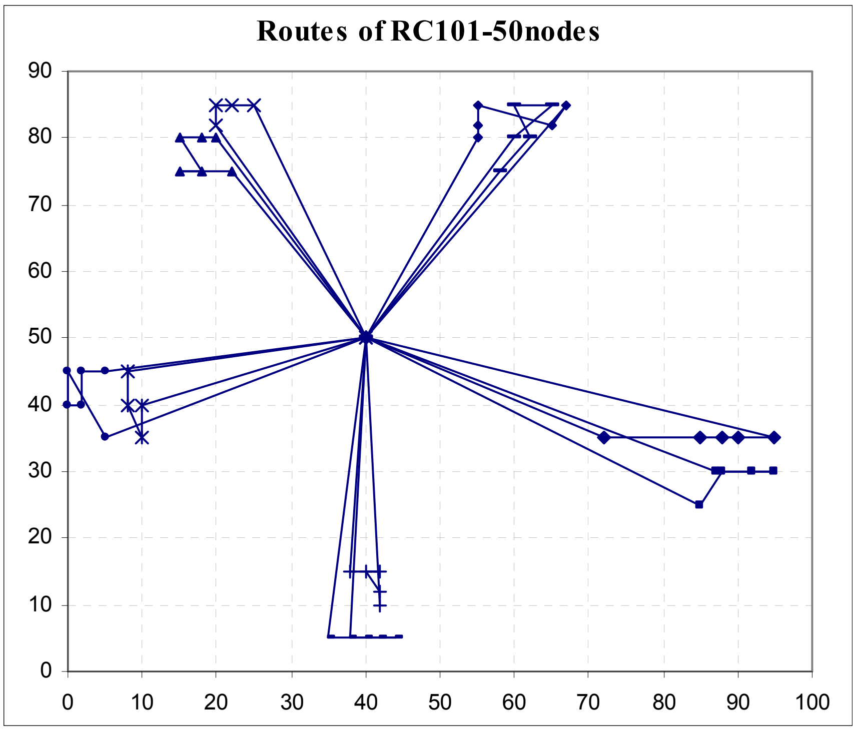


Figure I11.2: Routes of RC101-50 nodes

Table I11.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC101-25	8	1	121,07	0-5-2-7-6-8-3-1-4-0
	8	2	184,37	0-22-20-0
				0-23-21-19-18-25-24-0
	9	2	176,84	0-12-11-9-10-0
				0-14-15-16-17-13-0
RC101-50	10	2	238,74	0-27-29-31-34-50-0
				0-33-30-28-26-32-0
	10	2	171,51	0-2-7-6-8-46-4-0
				0-45-5-3-1-0
	10	2	179,01	0-12-11-9-10-0
				0-14-47-15-16-17-13-0
	10	2	192,03	0-19-49-22-20-24-0
				0-23-21-18-48-25-0
	10	2	187,51	0-39-36-40-38-41-0
				0-42-44-43-37-35-0

I12. RC102

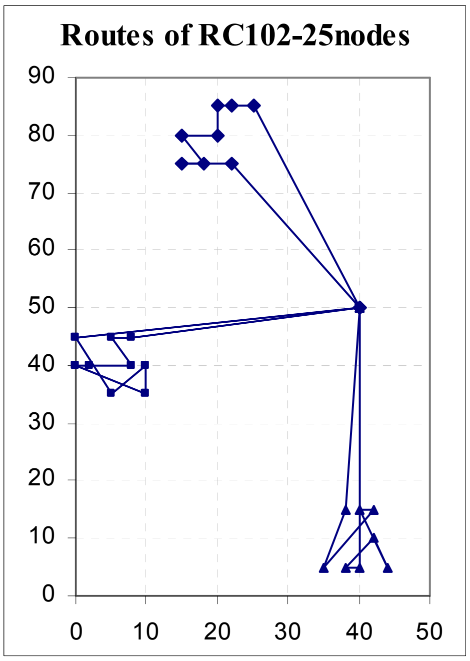


Figure I12.1: Routes of RC102-25 nodes

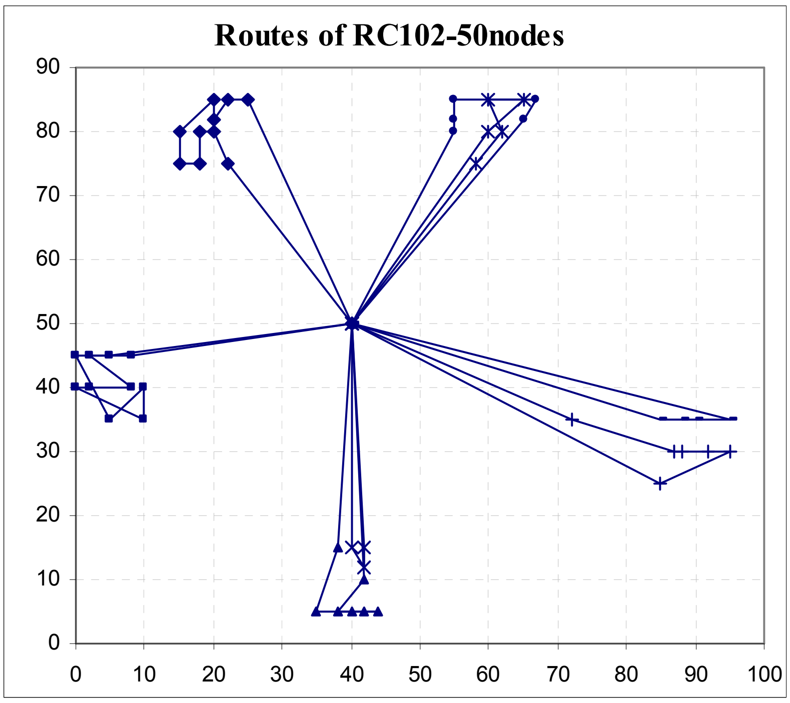


Figure I12.2: Routes of RC102-50 nodes

Table I12.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC102-25	8	1	129,26	0-21-23-19-18-22-20-25-24-0
	9	1	123,96	0-12-14-11-15-16-9-10-13-17-0
	8	1	99,72	0-2-7-6-8-4-5-3-1-0
RC102-50	10	1	105,95	0-1-3-45-5-8-7-6-46-4-2-0
	10	1	128,99	0-14-47-11-15-16-9-10-13-17-12-0
	10	2	236,254	0-33-26-28-30-32-50-0
				0-34-31-29-27-0
	10	2	183,6131	0-19-23-48-18-21-25-24-0
				0-22-49-20-0
	10	2	185,4752	0-39-36-40-38-41-0
				0-42-44-43-35-37-0

I13. RC103

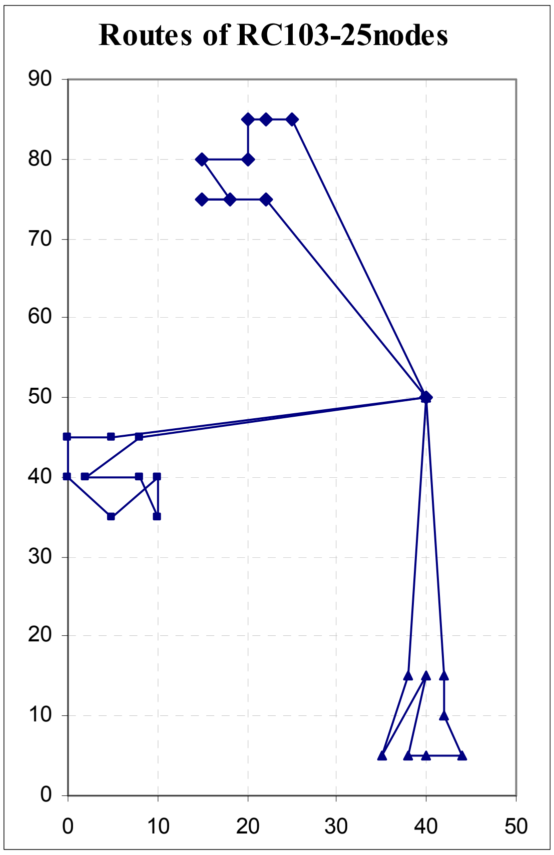


Figure I13.1: Routes of RC103-25 nodes

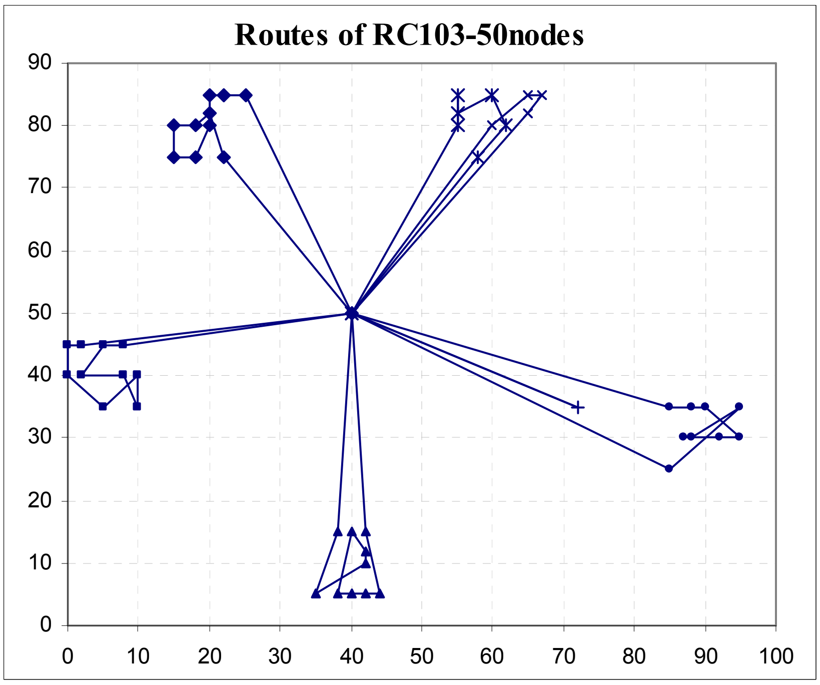


Figure I13.2: Routes of RC103-50 nodes

Table I13.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC103-25	8	1	99,7157	0-2-7-6-8-4-5-3-1-0
	9	1	116,0812	0-12-15-11-9-10-13-16-17-14-0
	8	1	118,318	0-20-19-18-21-23-22-25-24-0
RC103-50	10	2	213,4102	0-33-27-30-32-28-26-29-31-34-0
				0-50-0
	10	2	179,3061	0-39-36-35-37-0
				0-42-43-44-40-38-41-0
	10	1	105,3784	0-2-45-46-8-7-6-4-5-3-1-0
	10	1	117,0741	0-12-14-15-11-9-10-13-16-17-47-0
	10	1	121,15845	0-20-18-48-21-23-22-49-19-25-24-0

I14. RC104

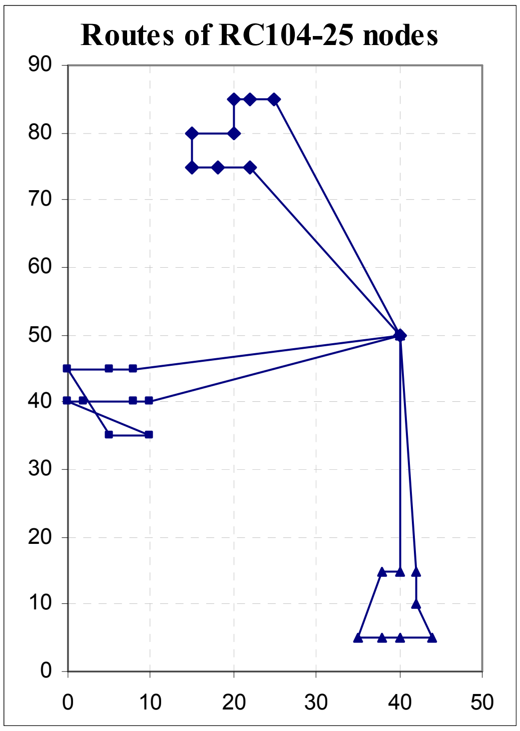


Figure I14.1: Routes of RC104-25 nodes

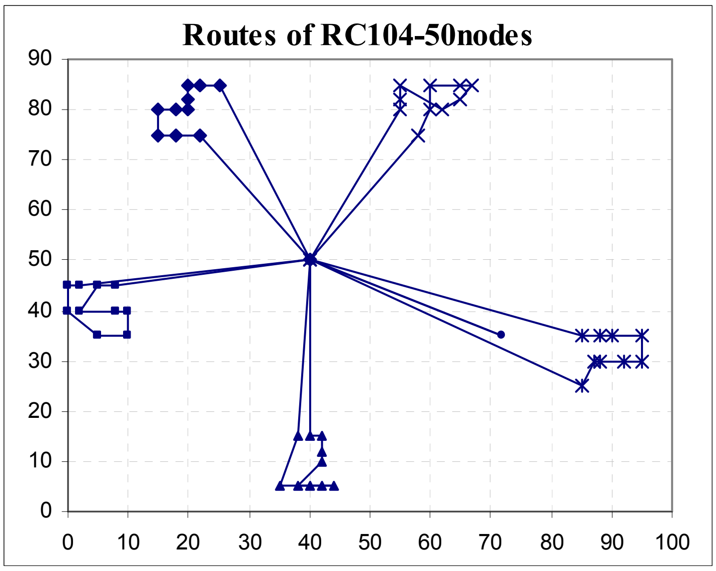


Figure I14.2: Routes of RC104-50 nodes

Table I14.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC104-25	8	1	95,8847	0-2-6-7-8-4-5-3-1-0
	9	1	109,3717	0-10-11-15-16-9-13-17-14-12-0
	8	1	101,8826	0-20-19-18-21-23-25-24-22-0
RC104-50	10	2	197,9799	0-34-31-29-27-26-28-30-32-33-0
				0-50-0
	10	1	95,8847	0-2-6-7-8-46-4-45-5-3-1-0
	10	1	111,6178	0-12-14-15-11-10-9-13-16-17-47-0
	10	1	108,9005	0-22-20-49-19-23-21-18-48-25-24-0
	10	1	102,5455	0-42-44-43-38-37-35-36-40-39-41-0

I15. RC105

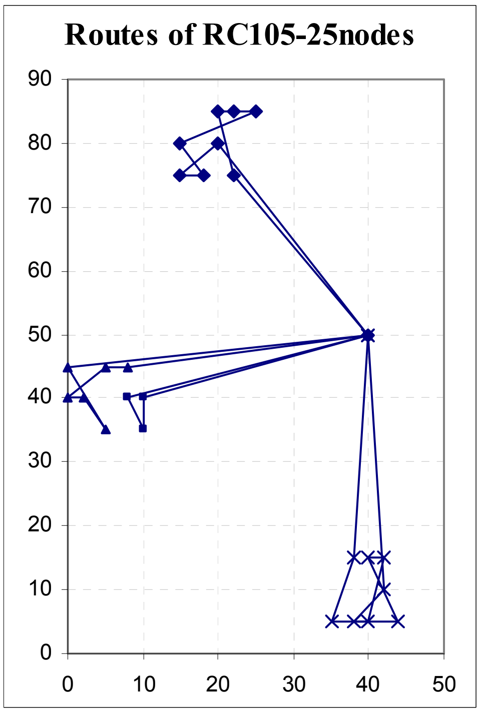


Figure I15.1: Routes of RC105-25 nodes

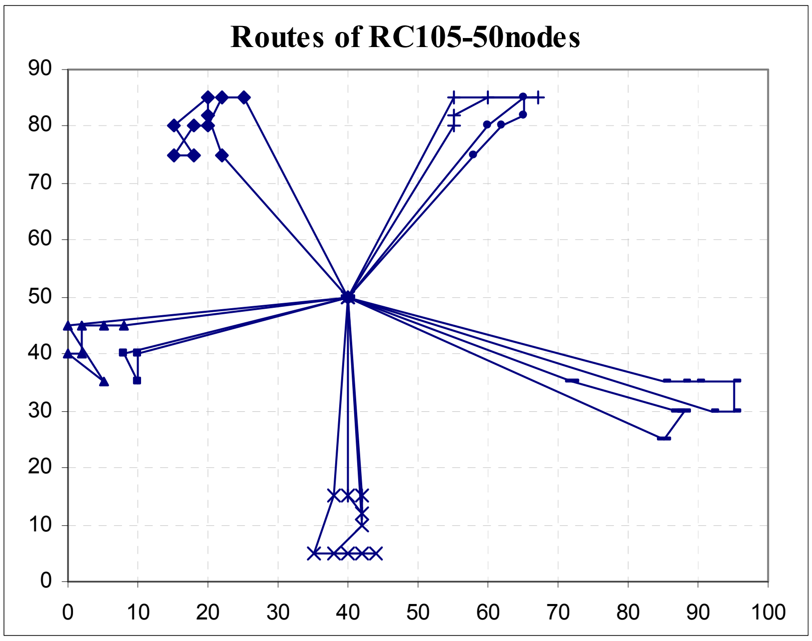


Figure I15.2: Routes of RC105-50 nodes

Table I15.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC105-25	8	1	109,1418	0-2-5-3-1-8-6-7-4-0
	8	2	177,316	0-11-9-10-0
				0-12-14-16-15-13-17-0
	9	1	125,9189	0-19-23-18-22-20-21-25-24-0
RC105-50	10	2	230,6094	0-28-26-27-29-31-34-0
				0-33-30-32-50-0
	10	1	111,283	0-2-45-5-8-6-7-46-4-3-1-0
	10	2	179,485	0-11-9-10-0
				0-12-14-47-15-16-13-17-0
	10	2	183,6131	0-19-23-48-18-21-25-24-0
				0-22-49-20-0
	10	2	185,3919	0-39-36-37-38-41-0
				0-42-44-40-35-43-0

I16. RC106

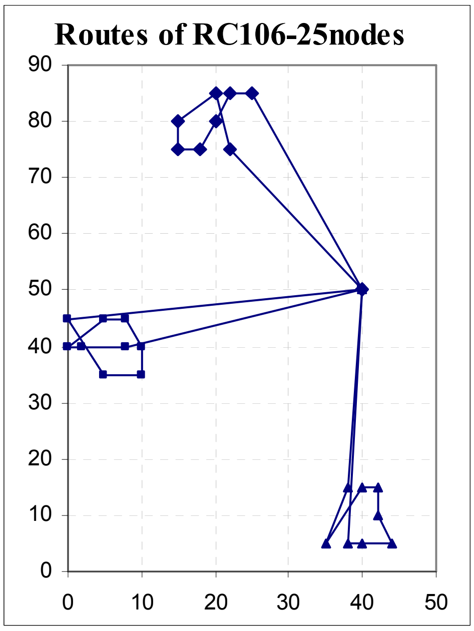


Figure I16.1: Routes of RC106-25 nodes

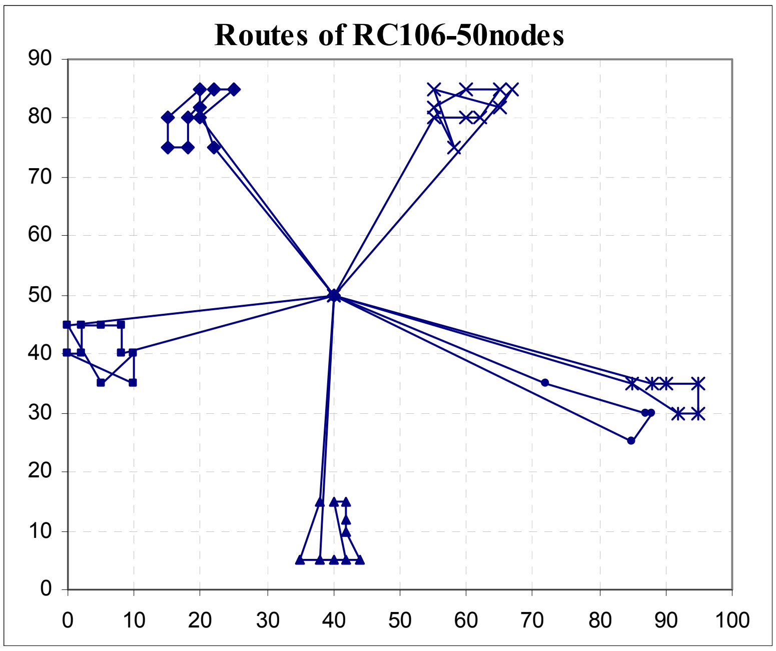


Figure I16.2: Routes of RC106-50 nodes

Table I16.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC106-25	8	1	107,9241	0-2-5-8-7-6-4-3-1-0
	8	1	118,474	0-11-15-16-14-12-10-9-13-17-0
	9	1	120,1073	0-23-21-18-19-20-22-25-24-0
RC106-50	10	2	230,7873	0-31-29-27-26-28-34-0
				0-33-30-32-50-0
	10	1	113,6867	0-2-45-5-8-7-6-46-3-1-4-0
	10	1	126,2691	0-11-12-14-47-15-16-9-10-13-17-0
	10	1	126,125	0-23-21-18-19-49-20-22-48-25-24-0
	10	1	133,5089	0-42-39-38-36-40-44-41-43-37-35-0

I17. RC107

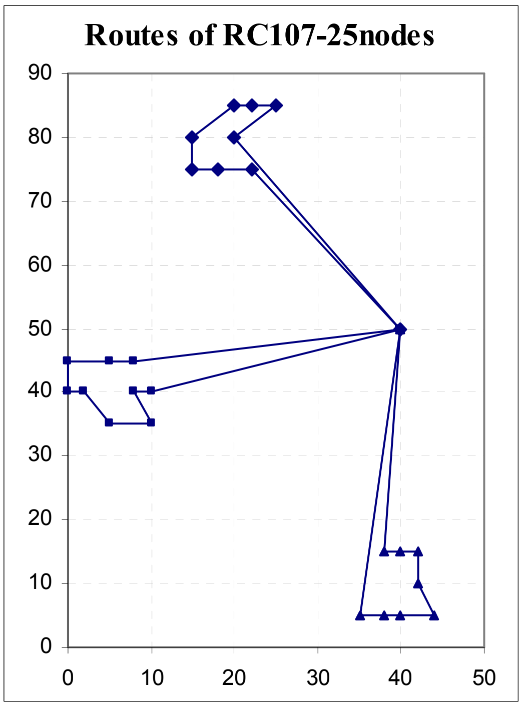


Figure I17.1: Routes of RC107-25 nodes

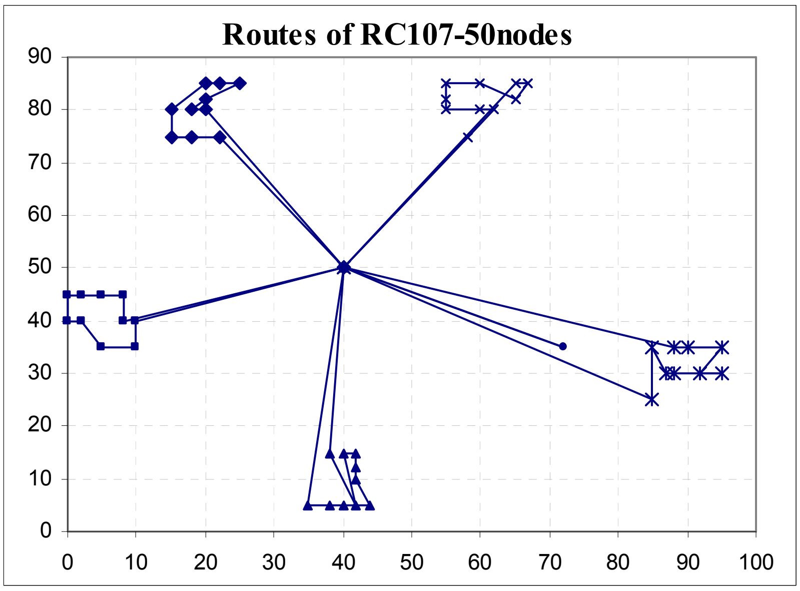


Figure I17.2: Routes of RC107-50 nodes

Table I17.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC107-25	8	1	98,0035	0-2-6-7-8-5-3-1-4-0
	9	1	97,2272	0-12-14-17-16-15-13-9-11-10-0
	8	1	103,7192	0-25-23-21-18-19-20-22-24-0
RC107-50	10	2	211,6658	0-31-29-27-28-26-30-32-34-33-0
				0-50-0
	10	1	101,5918	0-2-6-7-8-5-3-1-45-46-4-0
	10	1	100,9798	0-11-12-14-47-17-16-15-13-9-10-0
	10	1	122,6876	0-25-23-21-18-19-49-20-22-48-24-0
	10	1	108,6571	0-41-38-39-42-44-43-40-37-35-36-0

I18. RC108

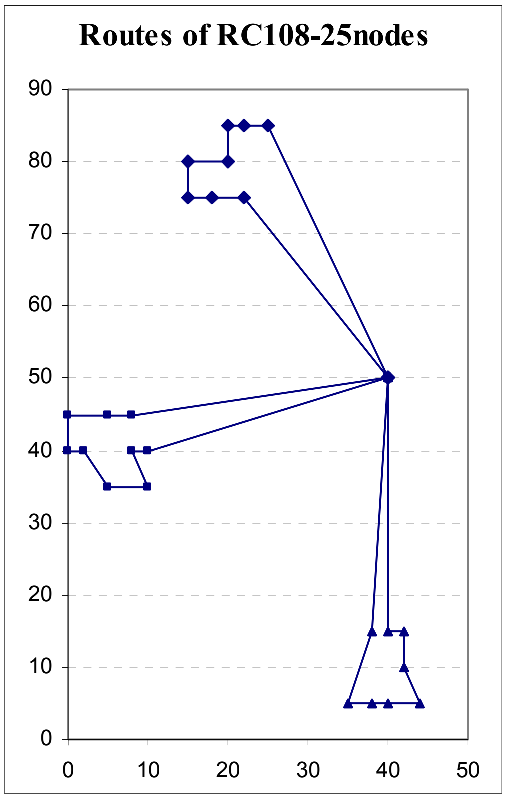


Figure I18.1: Routes of RC108-25 nodes

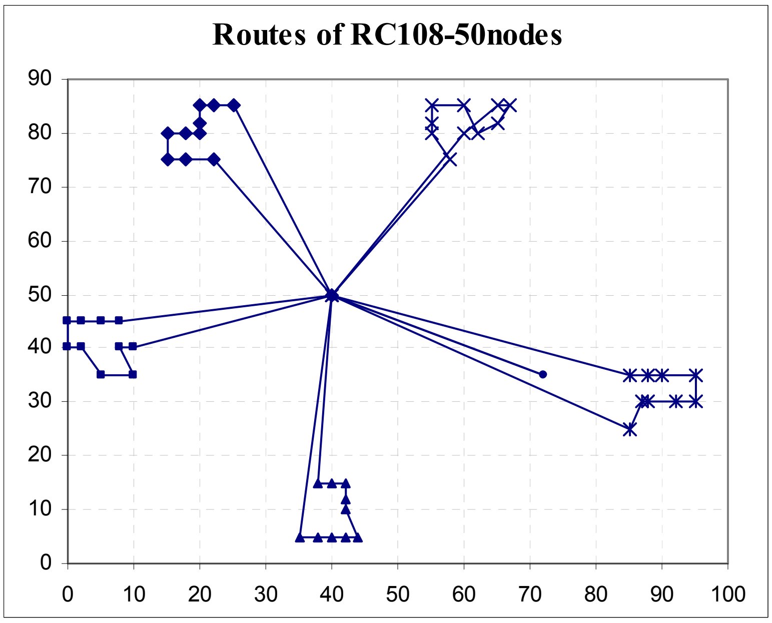


Figure I18.2: Routes of RC108-50 nodes

Table I18.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC108-25	8	1	95,88	0-2-6-7-8-4-5-3-1-0
	9	1	97,23	0-12-14-17-16-15-13-9-11-10-0
	8	1	101,88	0-22-20-19-18-21-23-25-24-0
RC108-50	10	2	197,98	0-33-32-30-28-26-27-29-31-34-0
				0-50-0
	10	1	95,88	0-2-6-7-8-46-4-45-5-3-1-0
	10	1	97,23	0-12-14-47-17-16-15-13-9-11-10-0
	10	1	103,72	0-25-23-21-48-18-19-49-20-22-24-0
	10	1	104,36	0-41-42-44-43-40-38-37-35-36-39-0

I19. RC201

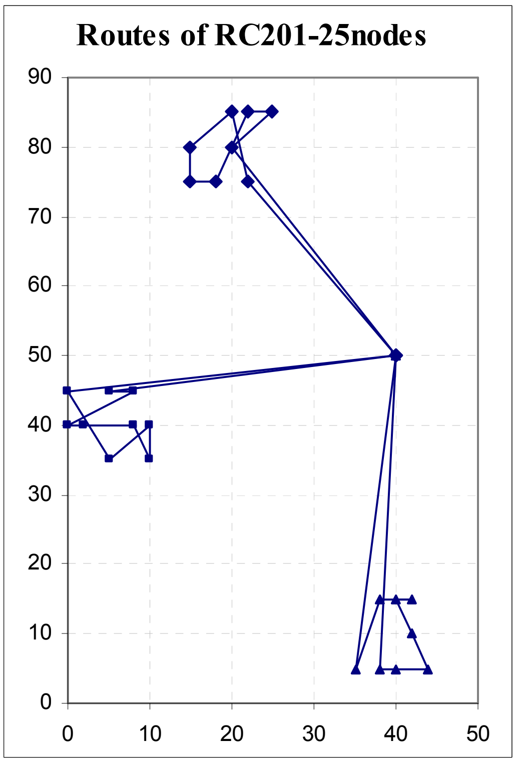


Figure I19.1: Routes of RC201-25 nodes

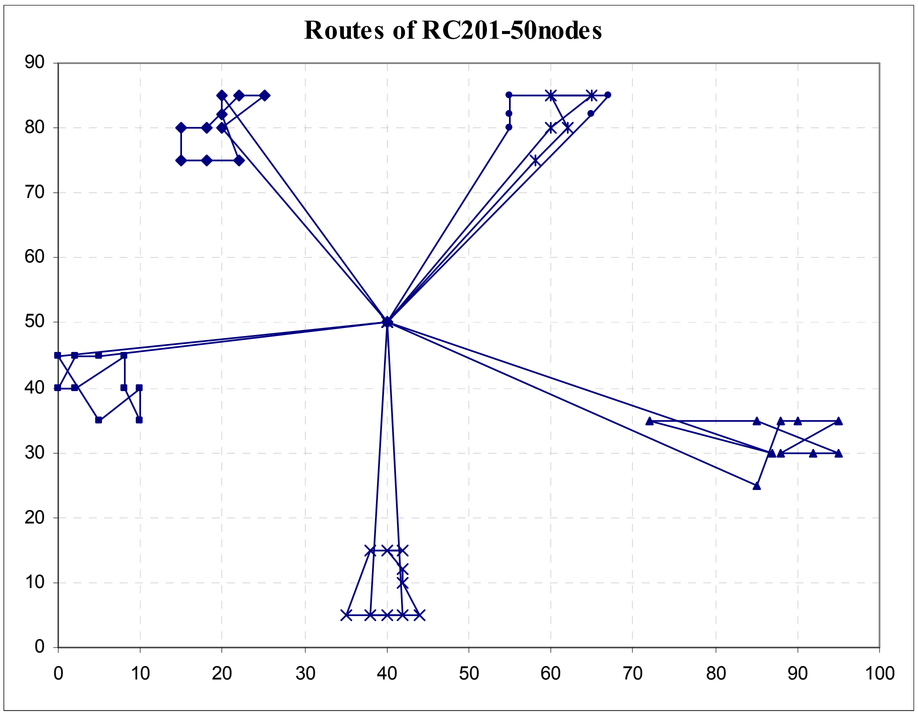


Figure I19.2: Routes of RC201-50 nodes

Table I19.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC201-25	8	1	112,97	0-2-5-8-7-6-3-1-4-0
	9	1	124,74	0-14-12-16-15-11-9-10-13-17-0
	8	1	123,53	0-23-21-18-19-22-20-24-25-0
RC201-50	10	1	118,21	0-5-45-2-6-7-8-46-3-1-4-0
	10	1	127,50	0-14-47-16-15-12-11-9-10-13-17-0
	10	1	130,52	0-23-21-18-19-49-22-20-24-25-48-0
				0-39-36-40-38-41-0
	10	2	185,48	0-42-44-43-35-37-0
	10	1	175,59	0-33-31-29-27-30-28-26-34-50-32-0

I20. RC202

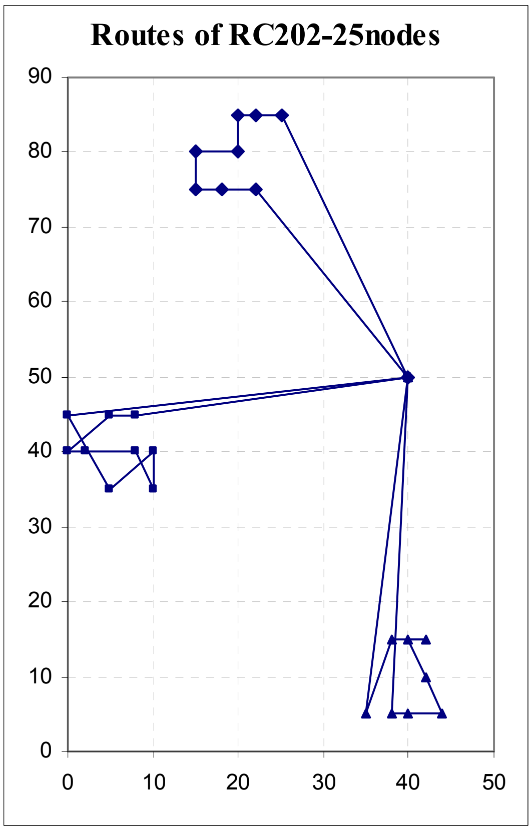


Figure I20.1: Routes of RC202-25 nodes

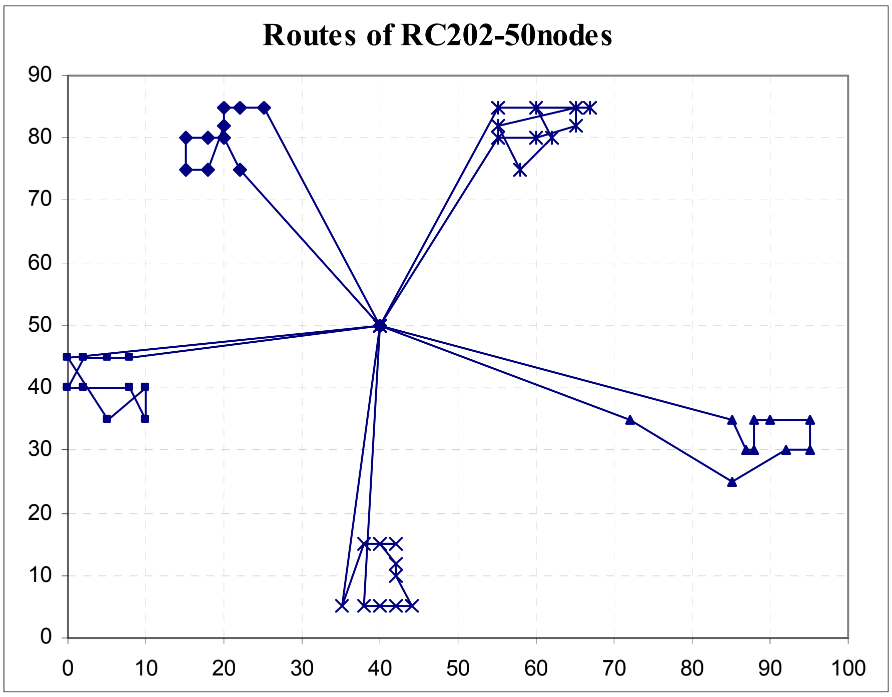


Figure I20.2: Routes of RC202-50 nodes

Table I20.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC202-25	8	1	95,88	0-2-6-7-8-4-5-3-1-0
	9	1	119,41	0-12-14-16-15-11-9-10-13-17-0
	8	1	123,53	0-23-21-18-19-22-20-24-25-0
RC202-50	10	1	102,55	0-1-3-5-45-6-7-8-46-4-2-0
	10	1	120,72	0-12-14-47-16-15-11-9-10-13-17-0
	10	1	123,75	0-23-21-48-18-19-49-22-20-24-25-0
	10	1	133,85	0-42-39-37-36-44-41-38-40-35-43-0
	10	1	134,16	0-50-33-28-26-27-29-31-30-32-34-0

I21. RC203

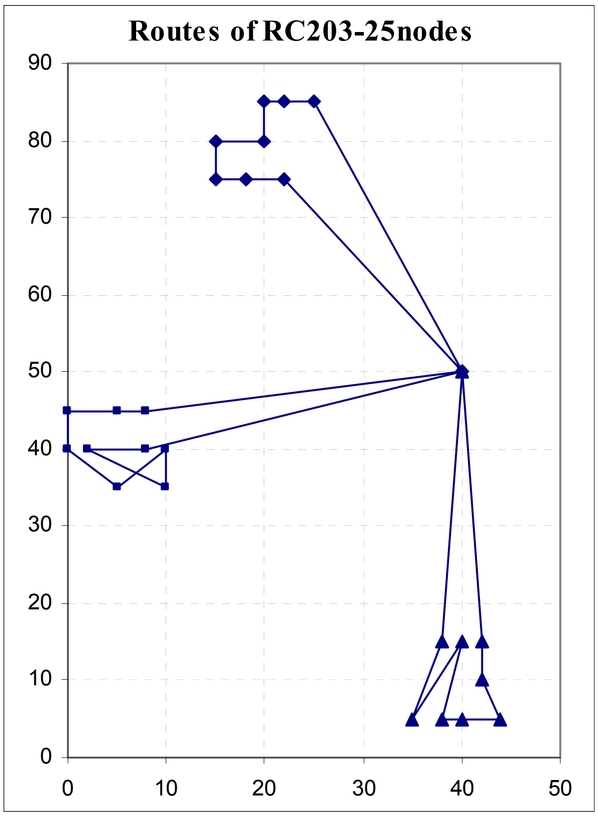


Figure I21.1: Routes of RC203-25 nodes

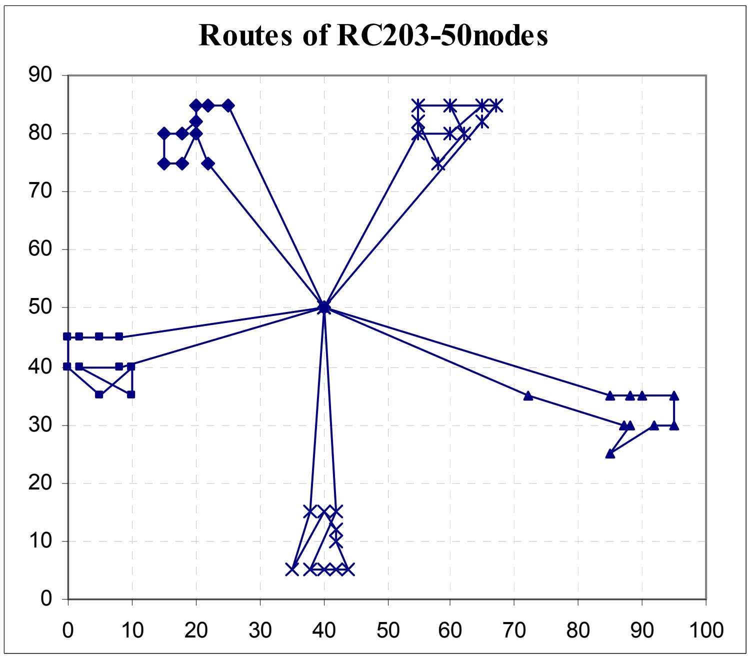


Figure I21.2: Routes of RC203-50 nodes

Table I21.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC203-25	8	1	95,88	0-2-6-7-8-4-5-3-1-0
	8	1	113,49	0-11-15-9-10-13-16-17-14-12-0
	9	1	118,32	0-20-19-18-21-23-22-25-24-0
RC203-50	10	1	101,48	0-1-3-5-45-46-8-7-6-4-2-0
	10	1	113,49	0-11-15-9-10-13-16-17-47-14-12-0
	10	1	119,50	0-20-23-21-48-18-19-49-22-25-24-0
	10	1	129,23	0-42-39-36-43-44-41-38-40-35-37-0
	10	1	132,02	0-34-31-29-27-26-28-33-30-32-50-0

I22. RC204

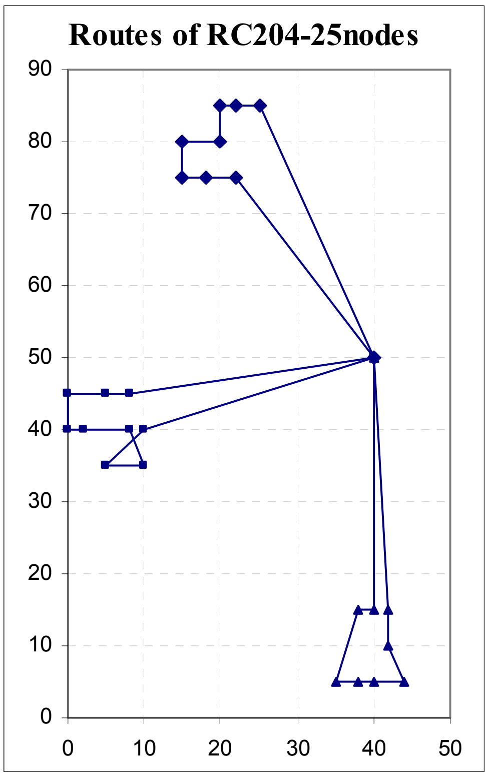


Figure I22.1: Routes of RC204-25 nodes

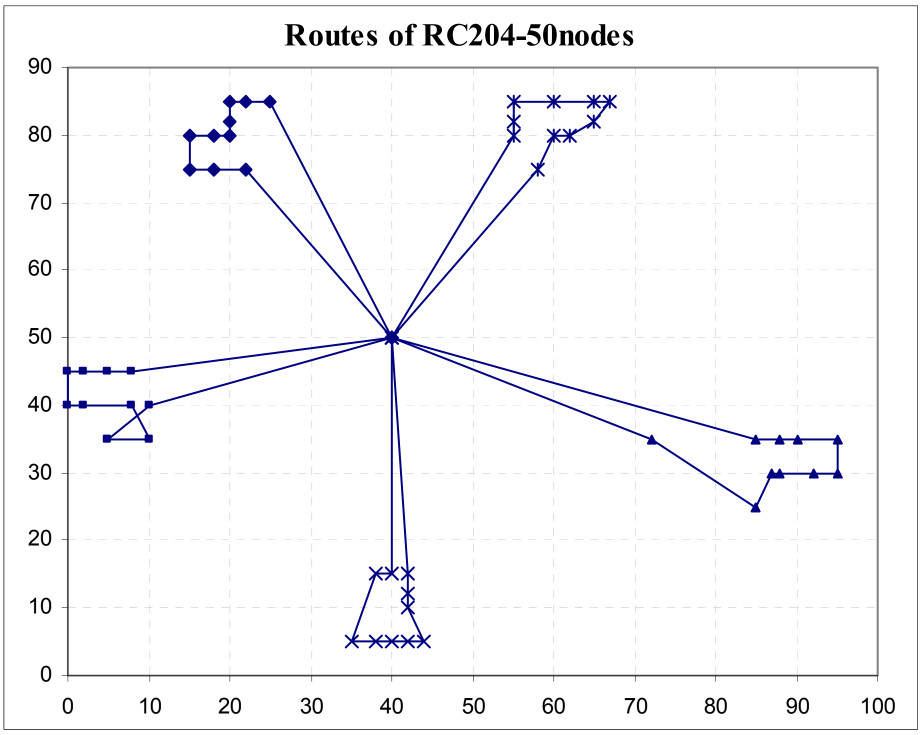


Figure I22.2: Routes of RC204-50 nodes

Table I22.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC204-25	8	1	95,88	0-2-6-7-8-4-5-3-1-0
	9	1	102,47	0-12-14-17-16-15-11-9-13-10-0
	8	1	101,88	0-20-19-18-21-23-25-24-22-0
RC204-50	10	1	95,88	0-2-6-7-8-46-4-45-5-3-1-0
	10	1	102,47	0-12-14-47-17-16-15-11-9-13-10-0
	10	1	101,88	0-20-49-19-18-48-21-23-25-24-22-0
	10	1	95,94	0-42-44-43-40-36-35-37-38-39-41-0
	10	1	127,56	0-50-33-32-30-28-26-27-29-31-34-0

I23. RC205

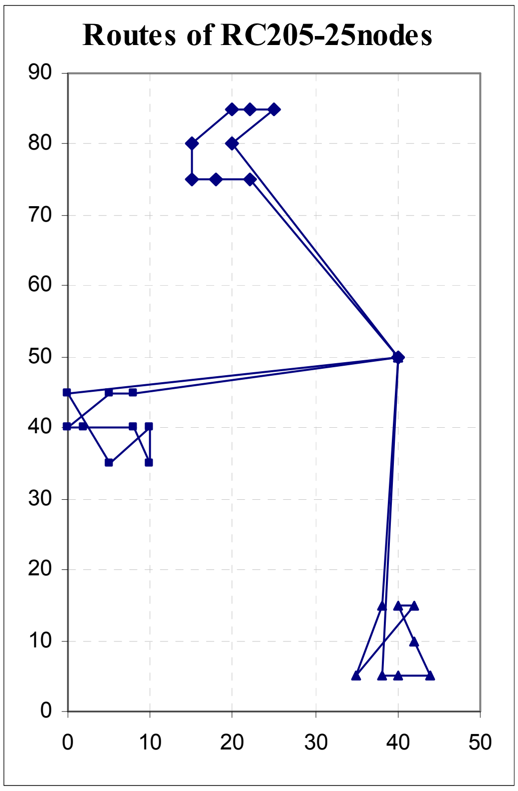


Figure I23.1: Routes of RC205-25 nodes

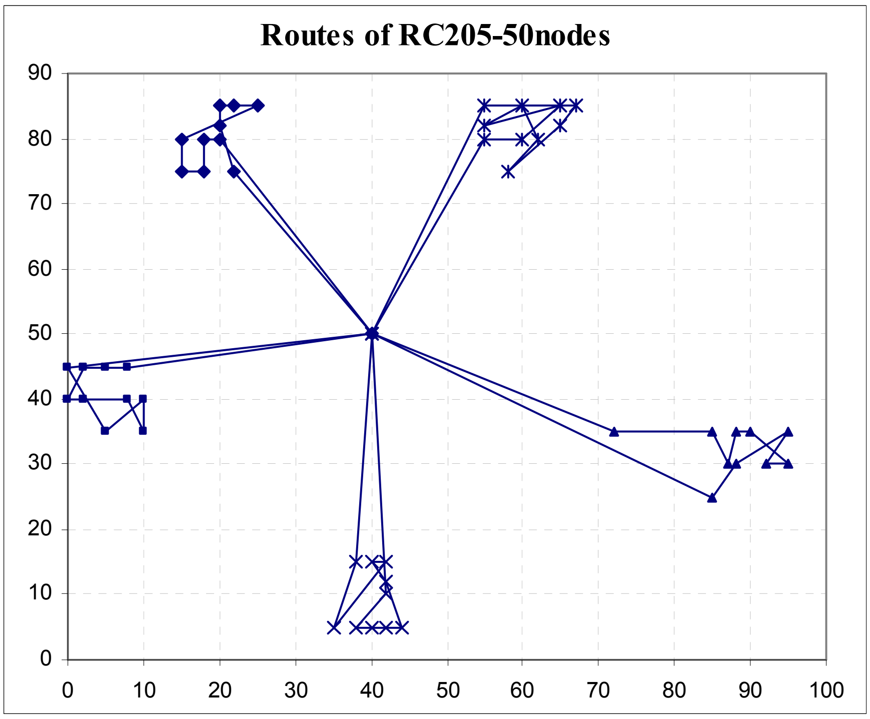


Figure I23.2: Routes of RC205-50 nodes

Table I23.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC205-25	8	1	98,00	0-2-6-7-8-5-3-1-4-0
	9	1	119,41	0-12-14-16-15-11-9-10-13-17-0
	8	1	121,52	0-23-21-18-19-22-20-25-24-0
RC205-50	10	1	108,32	0-2-45-5-3-1-8-7-6-46-4-0
	10	1	120,72	0-12-14-47-16-15-11-9-10-13-17-0
	10	1	123,04	0-19-23-21-48-18-49-22-20-25-24-0
	10	1	137,26	0-42-39-36-44-40-38-41-37-35-43-0
	10	1	142,64	0-33-30-27-28-26-29-31-32-34-50-0

I24. RC206

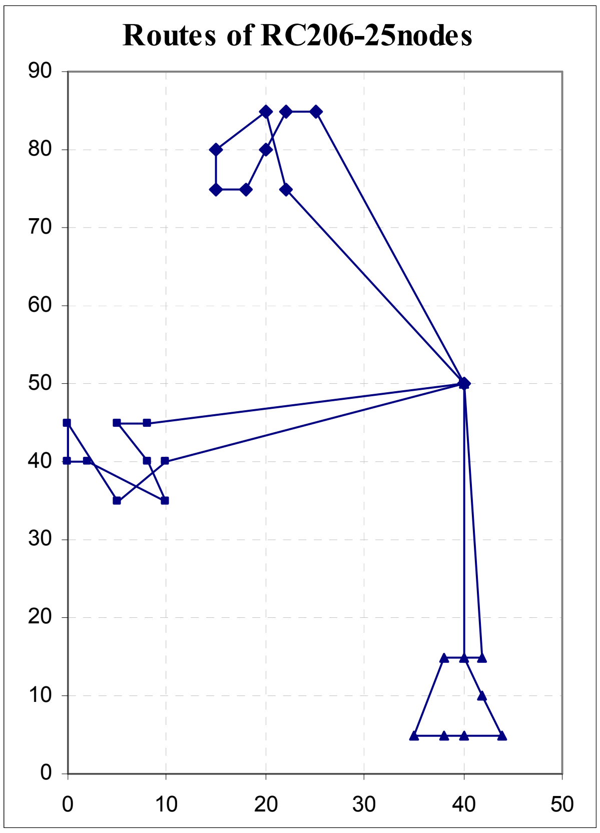


Figure I24.1: Routes of RC206-25 nodes

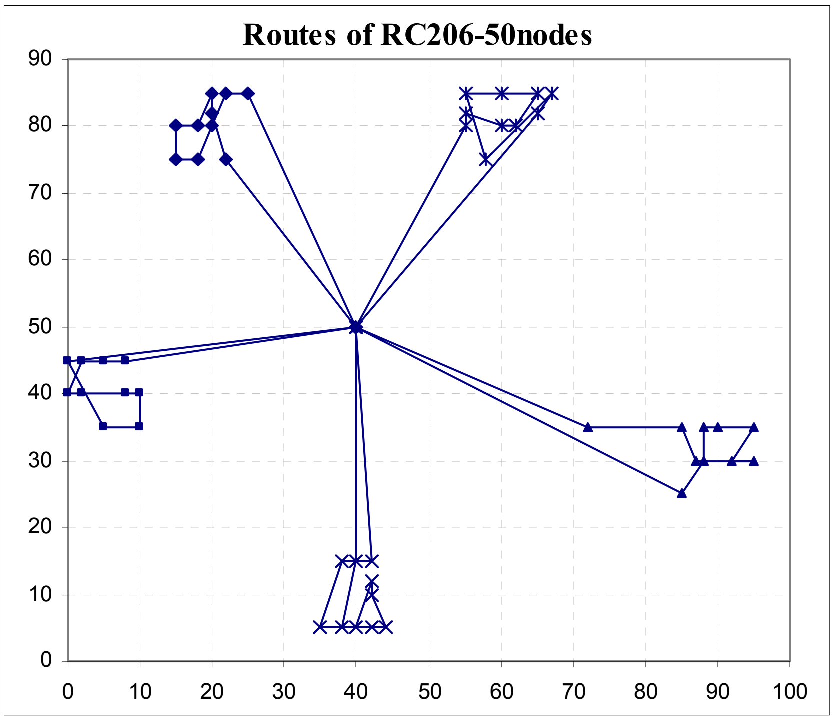


Figure I24.2: Routes of RC206-50 nodes

Table I24.1: Solutions of subproblems

Problem	**Number of Nodes (n)**	**Number of Vehicles (v)**	**Traveling Cost**	**Route**
RC206-25	8	1	107,92	0-2-5-8-7-6-4-3-1-0
	9	1	112,91	0-12-14-11-9-15-16-17-13-10-0
	8	1	104,27	0-22-19-18-21-23-25-24-20-0
RC206-50	10	1	109,32	0-2-45-5-46-8-7-6-4-3-1-0
	10	1	115,27	0-12-14-47-16-15-11-10-9-13-17-0
	10	1	120,36	0-22-23-21-49-19-18-48-25-24-20-0
	10	1	126,86	0-42-44-39-38-36-40-43-41-35-37-0
	10	1	139,87	0-33-30-31-29-27-28-26-32-34-50-0

I25. RC207

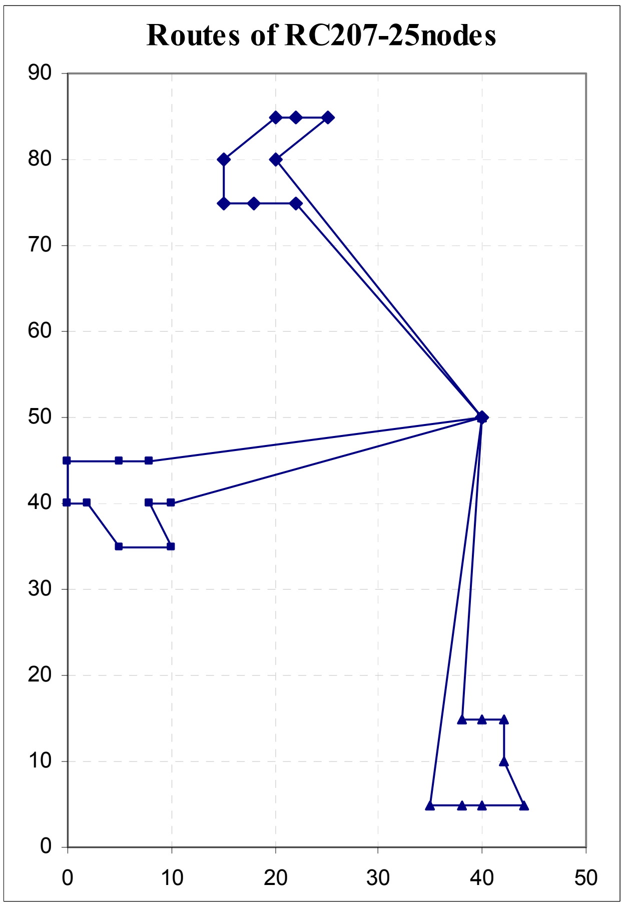


Figure I25.1: Routes of RC207-25 nodes

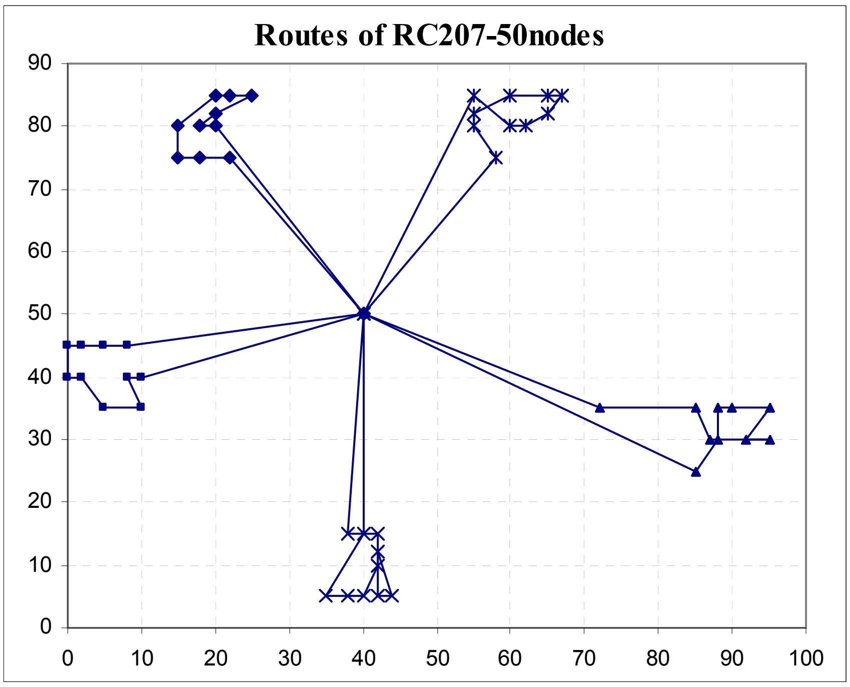


Figure I25.2: Routes of RC207-50 nodes

Table I25.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC207-25	8	1	98,00	0-2-6-7-8-5-3-1-4-0
	9	1	97,23	0-12-14-17-16-15-13-9-11-10-0
	8	1	103,72	0-25-23-21-18-19-20-22-24-0
RC207-50	10	1	101,59	0-2-6-7-8-5-3-1-45-46-4-0
	10	1	97,23	0-12-14-47-17-16-15-13-9-11-10-0
	10	1	116,90	0-22-25-23-21-19-49-18-48-20-24-0
	10	1	105,83	0-41-42-44-40-36-35-37-38-39-43-0
	10	1	139,87	0-33-30-31-29-27-28-26-32-34-50-0

I26. RC208

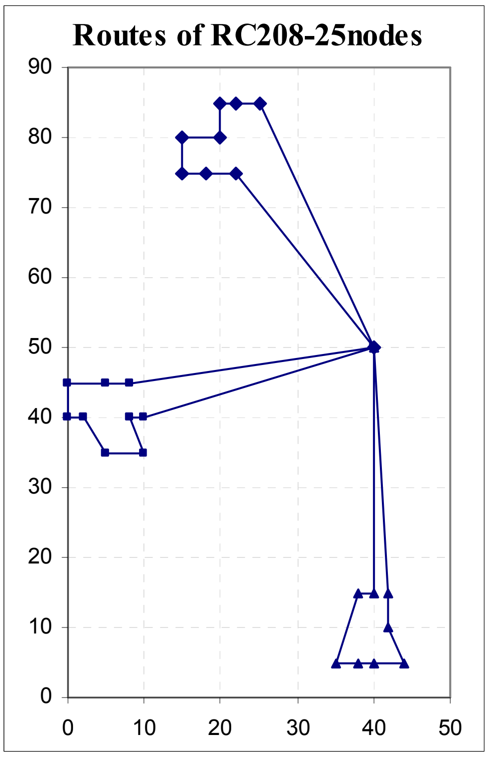


Figure I26.1: Routes of RC208-25 nodes

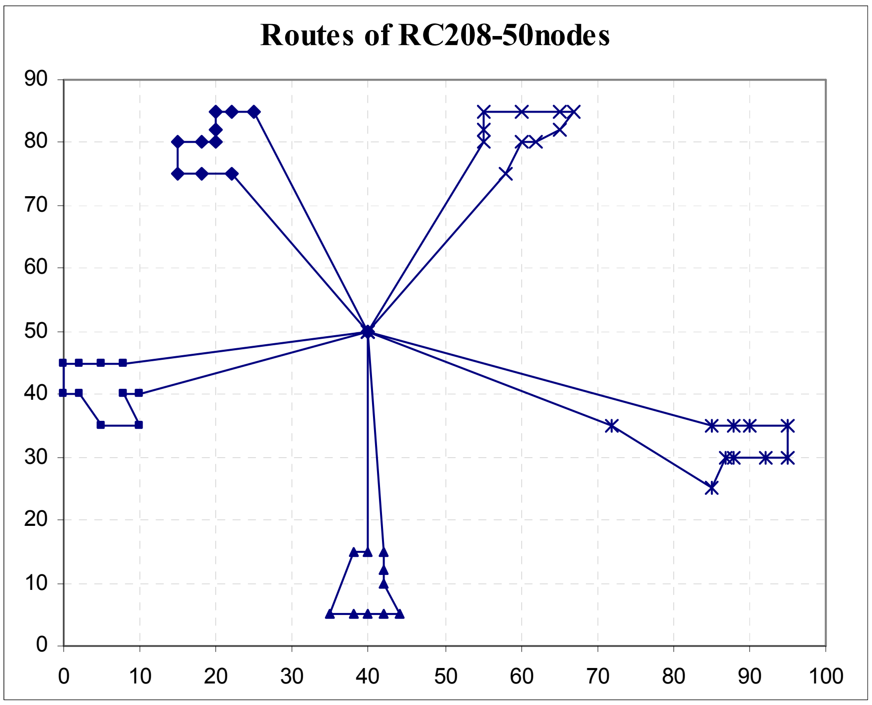


Figure I26.2: Routes of RC208-50 nodes

Table I26.1: Solutions of subproblems

Problem	Number of Nodes (n)	Number of Vehicles (v)	Traveling Cost	Route
RC208-25	8	1	95,88	0-1-3-5-4-8-7-6-2-0
	9	1	97,23	0-12-14-17-16-15-13-9-11-10-0
	8	1	101,88	0-20-19-18-21-23-25-24-22-0
RC208-50	10	1	95,88	0-1-3-5-45-4-46-8-7-6-2-0
	10	1	97,23	0-12-14-47-17-16-15-13-9-11-10-0
	10	1	101,88	0-20-49-19-18-48-21-23-25-24-22-0
	10	1	95,94	0-42-44-43-40-36-35-37-38-39-41-0
	10	1	127,56	0-50-33-32-30-28-26-27-29-31-34-0