TOBB UNIVERSITY OF ECONOMICS AND TECHNOLOGY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

A FIX-AND-OPTIMIZE HEURISTIC FOR THE INTEGRATED FLEET SIZING AND REPLENISHMENT PLANNING PROBLEM WITH PREDETERMINED DELIVERY FREQUENCIES

M.sc. THESIS

Niousha KARIMI DASTJERD

Industrial Engineering Science Programme

Supervisor: Assoc. Prof. Dr. Kadir ERTOĞRAL

April 2016

Confirmation of Graduate School of Natural and Applied Sciences

……..
Prof. Dr. Osman EROĞUL
Director

I confirm that this thesis is submitted in partial fullfilment of the requirements for the degree of master of science.

…….
Prof. Dr. Tahir HANALIOĞLU
Head of Department

The thesis with the title of "**A FIX-AND-OPTIMIZE HEURISTIC FOR THE INTEGRATED FLEET SIZING AND REPLENISHMENT PLANNING PROBLEM WITH PREDETERMINED DELIVERY FREQUENCIES**" prepared by **Niousha KARIMI DASTJERD** M.sc./Ph.d. student of Natural and Applied Sciences institute of TOBB ETU with the student ID 131311023 has been approved on **11.04.2016** by the following examining committee members, after fulfillment of requirements specified by academic regulations.

Supervisor: Assoc. Prof. Dr. Kadir ERTOĞRAL .....
TOBB University of Economics and Technology

Examining committee members:

Prof. Dr. Fülya ALTIPARMAK (Head) .....
Gazi University

Assis. Prof. Dr. Ayşegül ALTIN KAYHAN .....
TOBB University of Economics and Technology

Assoc. Prof. Dr. Kadir ERTOĞRAL .....
TOBB University of Economics and Technology

THESIS STATEMENT

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Niousha KARIMI DASTJERD

Signatur

ABSTRACT

Master of Science

A FIX-AND-OPTIMIZE HEURISTIC FOR THE INTEGRATED FLEET SIZING AND REPLENISHMENT PLANNING PROBLEM WITH PREDETERMINED DELIVERY FREQUENCIES

Niousha KARIMI DASTJERD

TOBB University of Economics and Technology

Institute of Natural and Applied Sciences

Industrial Engineering Science Programme

Supervisor: Assoc. Prof. Dr. Kadir ERTOĞRAL

Date: April 2016

We tackled an integrated fleet sizing and replenishment planning problem in a vendor managed inventory system. There is a set of customers which must be replenished based on a given set of predetermined frequencies. The vehicle fleet consists of multiple types of heterogeneous vehicles which differ in carrying capacity, cost per kilometer, and ownership costs. Customer demands are taken as deterministic values. The main decision we make in this problem is the triple asignment of vehicle-frequency-customer. As a result of these assignment decisions,

we obtain an annual costs consisting of vehichle ownership cost, routing cost, inventory holding and fixed replenishment costs. A key simplification in the model is the use of linear approxiamation for the routing cost based on the number of customers visited in a tour. The developed model, which is new in the literature, integrates fleet sizing and replenishment planning decisions.

Our problem is NP-hard since it can be shown that a special case of our problem is a bin packing problem. In order to solve large problems efficiently, we suggested and applied a fix and optimize heuristic as a solution procedure. This fix and optimize heuristic divides the problem into smaller problems in which some variables are binaries and the others are linearly relaxed, and it fixes the linear decision variable iteratively. We also showed the effectiveness of the suggested heuristic solution procedure on a large set of randomly generated problems.

Keywords: Fleet sizing, Replenishment planning, Predetermined frequencies, Fix and Optimize.

ÖZET

Yüksek Lisans

ÖNCEDEN BELİRLENMİŞ TESLİMAT FREKANSLARI İLE ENTEGRE FİLO BOYUTLANDIRMA VE İKMAL PLANLAMA PROBLEMİ İÇİN SABİTLE VE OPTİMİZE ET SEZGİSEL YÖNTEMİ UYGULANIŞI

Niousha KARİMİ DASTJERD

TOBB Ekonomi ve Teknoloji Üniveritesi

Fen Bilimleri Enstitüsü

Endüstri Mühendisliği

Danışman: Doç. Dr. Kadir ERTOĞRAL

Tarih: Nisan 2016

Bu tez çalışmasında satıcı yönetimli stok politikası uygulayan sistemler için filo büyüklüğü ve ikmal planlamasının entegre şekilde belirlenmesi ele alınmıştır. Önceden belirlenmiş frekans setine göre ikmal edilen müşteri seti mevcuttur. Araç filosu birden fazla farklı araçtan oluşmaktadır ve bu araçlar sabit kilometre başı maliyetler, taşıma kapasitesi ve edinme maliyetleri açısından farklılık arz etmekteler. Müşteri talepleri deterministik değerler olarak alınmıştır. Bu problemde verilen asıl karar araç- frekans – müşteri üçlüsünün atamasıdır. Bu atama kararları sonucunda, araç edinme maliyeti, rotalama maliyeti, envanter tutma maliyeti ve sabit ikmal yapma maliyetinden oluşan toplam maliyet elde edilmektedir. Bu

modeldeki en önemli basitleştirme, rotalama maliyetinin bir tur içerisinde ziyaret edilen müşterilerin sayısına bağlı olarak yaklaşık bir değer şeklinde kullanılmasıdır. Bu tez çalışmasında geliştirilen model literatürde yeni bir modeldir ve filo büyüklüğü belirleme ve ikmal planlaması kararlarını entegre şekilde vermektedir.

Bizim problem kutulama probleminin özel haline dönüşebilmesi nedeni ile NP-Zor bir problemdir. Uzun çözüm sürelerini ortadan kaldırmak amacıyla sabitle ve optimize et sezgiseli çözüm yöntemi olarak önerilip uygulanmıştır. Sabitle ve optimize et yöntemi ana problemi bazı değişkenleri ikili ve diğer değişkenleri doğrusal olarak gevşetilmiş küçük problemlere ayırmaktadır, ve doğrusal karar değişkenleri her iterasyonda sabitlenmektedir. Aynı zamanda, önerilen sezgisel yönteminin etkenliği rassal olarak üretilmiş büyük problem setlerine uygulanarak gösterilmiştir.

Anahtar kelime : Filo büyüklüğü belirleme, İkmal planlaması, Önceden belirlenmiş frekanslar, Sabitle ve Optimize et.

ACKNOWLEDGMENTS

First of all, I am deeply grateful to my supervisor Assoc. Prof. Dr. Kadir ERTOĞRAL for his invaluable supervision, guidance, criticism and especially his extreme support not only during this study but also in the entire period of my graduate study.

I gratefully acknowledge the scholarship received towards my M.Sc. from the TOBB University of Economics and Technology M.Sc. fellowship.

I also thank examining committee members, Prof. Dr. Fülya ALTIPARMAK and Assist. Prof. Dr. Ayşegül ALTIN KAYHAN for their valuable comments and contributions.

I would like to thank my lovely family for their support, love, and encouragement through all my life.

I would like to convey my deepest thanks to my husband, Reza AGHAZADEH, whose understanding and support gave me the motivation to pass this challenging period of my life.

CONTENTS

Page

TABLE OF CHARTS

Page

TABLE OF TABLES

Page

1. INTRODUCTION

Considering distribution systems, one of the key factors is the efficient transportation. According to Hoff et al. (2010) in general logistic costs constitute about 20% of the total cost of a product. Logistic costs can be reduced significantly by determining the fleet sizes efficiently since the owning and maintatining a fleet is a major cost componenet in total logistics cost for the firms that keeps a fleet.

Generally speaking, a fleet, is composed of heterogeneous vehicles. In the industry, the vehicle fleets are used for long periods of time, and in these period a fleet will gain different vehicles due to technological developments and market situations. During this period vehicle's maintenance, operation and depreciation costs will change. Another reason for preferring heterogeneous vehicle fleet is the operational constraints, and also the benefits which this flexibility provides inherently.

The vehicles distinguishing characteristics are divided into three categories:

- Physical dimension
- Compatibility constraints
- Costs

Physical dimensions such as length, height and width of a vehicle determine its carrying capacity. In road based transportations, sometimes physical dimensions obstruct reaching the route networks. As an example for this situation, we can mention the urban areas and narrow roads in villages, and limited space in ramps for loading and unloading. Dimension and weight constraints can vary during a given time horizon, as in seasonal axle pressure limits due to spring thaw. The vehicles speed, can also be categorized under physical dimensions class. Although vehicles with lower speed usually have lower unit cost, it is impossible to give less cost-efficient solutions due to temporal constraints.

Except physical dimensions, there exists another characteristic which limits vehicles utilization; “compatibility constraints”. These constraints can sometimes limit the places to which vehicles travel and also the loads which vehicles can carry. Often, customers need vehicles that have special equipment for loading and unloading operations. Operating vehicles in some areas necessitates special certificates. To illustrate this, we can mention the urban areas wherein fuel and noise emissions can limit vehicle utilization.

Another factor which affects vehicle fleet composition is the vehicle costs. Large vehicles usually have lower unit costs in comparison to small vehicles if their capacity is utilized effeciently. As mentioned before, when making decision about fleet compositions for a long period of time, the decision makers first of all have to make a strategic choice between renting and owning the vehicles in their fleet. Comparing the expected costs and incomes under uncertainty is one of the components which should be assessed before deciding about the fleet composition.

As it is an important part of the total cost of logistics, the cost related to inventory must also be controlled efficiently. Inventory cost is composed of holding cost, and fixed replenishment cost. In recent years, a business application called “Vendor Managed Inventory (VMI)” is applied by the firms which aim at minimizing the inventory costs. VMI systems are the systems in which the customer’s inventories are totally controlled by the vendor. Vendor decides when and how much to replenish each customer under the limits and constraints that customers determine, e.i. the minimum and maximum inventory level. VMI systems have many benefits on supplier’s side. As long as the supplier does his responsibility of controlling inventory levels and omitting stock outs, supplier can be locked into a VMI system supplier-customer relationship for a long period of time. In this way there will be a steady income for the supplier and the risk of supplier changes on the side of customer will be reduced. In a VMI system, the supplier has the chance of planning his operations more efficiently due to the ability of monitoring customer’s inventory levels in a steady manner. Additionally, during this monitoring procedure the suppliers have a much better understanding of the customer’s consumption rates during a given period of time which will cause decreases in stock amounts. Planning

customer's replenishments actually means that the supplier is determining the orders he faces largely by himself, and thus can make more effective production and replenishment decisions. In addition to all these benefits, a VMI system affords the supplier with the last customer's demand instead of the filtered demand information provided by the firms and thus can make better demand forecasts.

In supply chains, demand variability increases as moving downwards the chain. In the literature this phenomenon is called "bullwhip effect". Bullwhip phenomenon occurs due to material and information flows which are not done on the right time and with the right quantity throughout the supply chain. According to Hohmann and Zelewski (2012) another definition of bullwhip effect is the increase in variability of orders as a result of increase in customer demands. As illustrated formerly, in such a situation the flow of information and material along the supply chain layers will not be steadily. When bullwhip effect occurs, the orders look as if they have been hit by a whip and are fluctuated. This fluctuation in orders leads to higher inventory held in the warehouses and also quick responses to customer orders will not be guaranteed anymore. The factors which are effective in appearance of bullwhip effect are generally categorized as five groups: demand distortion, feedback misunderstandings, batch ordering, cost fluctuations and strategic behaviors. VMI systems are able to decrease the bullwhip effect in the supply chains to which they are applied due to their ability of allaying the demand distortion and feedback misunderstandings.

Considering transportation costs, VMI provides several benefits for the suppliers in the cases that customers are highly dispersed through various geographic regions. VMI makes it possible for the suppliers to send customer orders in consolidated form which will cause more efficient utilization of vehicles and as a result declines in transportation costs are met.

On the customer's side, the most important aspect of VMI application is the decline in inventory holding costs. In many situations the customers make a payment of the amounts which they have sold and are not forced to pay the opportunity type costs. Furthermore, the application of VMI yields elimination of other inventory management related costs such as labor costs, supervisory costs on customer's side.

In this thesis, we tackle a problem of simultaneously determining replenishment plans and fleet sizes. As mentioned in the literature, determining fleet composition needs solving vehicle routing problems. Finding optimal solution to vehicle routing problems requires a set of exact data about the locations of the customers, and leads into long solution times. We did not include exact routing in our modeling, which simplified the model drastically. There are several reasons for this simplification; First, the routing costs constitute relatively small portion of the total cost we have in our model and including routing cost in an aproximate fashion would not change final decisions significantly. Second, our model is a startegic model which considers routing cost in an approximate fashion since the exact routing and its costs would change from day to the next in a real problem. In the strategic type of problems the route length/ cost is assumed to be an approximate value in the literature as we point out in the literature part below. In general, the most of the studies in literature consider operational routing problems and to the best of our knowledge there are some papers which study strategic fleet sizing models. Among these fleet sizing models, none of them considers predetermined delivery frequencies and inventory related costs in a single model, which is the setup of our problem.

As a solution approach to our problem we suggested a " fix and optimize" heuristic. In this approach the main problem is divided into smaller sub problems. At each iteration, decision variables of one sub problem are defined as integers and the variables for other sub problems are linearly relaxed. After finding a full integer solution through utilization of this pattern, an improvement stage is executed.

This thesis is organized as below: first we will define our problem and analyze the complexity of the problem we have considered. Next the relevant literature is reviewed in chapter 3. After reviewing literature we will present numerical analysis of the problem in chapter 4. In chapter 5, valid inequalitis and lower bound analysis is illustrated. In chapters 6 and 7, we will present the heuristic we suggested and investigate the performance of the suggested heuristic. At the end, we summarize our work and contribution.

2. PROBLEM DESCRIPTION AND FORMULATION

In this section we formally state the problem we tackle. Basically, in our problem setup we aim at minimizing sum of the transportation and the inventory cost of shipping a product from one origin to multiple customer destinations. Mainly, by bringing solution to the problem under study, we intend to decide how often and how much to replenish each customer along with the determination of the composition of vehicle fleet. A single product with deterministic demand is made available at the origin and the product is demanded at multiple destinations with a constant rate. A set of heterogeneous vehicles is used to ship the product to the customers. Vehicles vary in different aspects such as carrying capacity, cost per kilometer, and ownership costs. Replenishments are carried out based on a set of given frequencies, which are, in general, on weekly basis. To exemplify, a customer can be replenished weekly, biweekly, thrice or quarto-weekly on any day of the week. Additionally, we can also have a daily frequency. In total, we have 21 different possible frequencies (4*5+1=21).

We are mainly concerned with determining the fleet size and the planning of the deliveries to the customers. To be more precise, the decisions our model will make are the number of each type of vehicle in the fleet of the vendor, along with the frequency, and the vehicle type with which each customer should be served.

Our objective function reflects a total annual cost and it is composed of two different types of costs; transportation related and inventory related costs. Transportation related costs consist of vehicle ownership costs and routing costs. Inventory related costs are fixed replenishment costs at the customers and inventory holding costs. All of these cost components are in the form of annual cost.

We also made some assumption regarding operational issues in our problem. Each customer must be served using a single vehicle and a single frequency. This assumption is in line with practice as customers would prefer regularity of single

frequency in delivery. Considering real life situations, daily traffics, and long distances, we assume that each vehicle can make one route in a day and there is a constraint on the number of customers a vehicle can visit on its daily route. We also take into account the fact that customers can be grouped geographically, and two customers from different geographic regions may not be on the same route.

An important aspect of our model is that it does not involve detailed routing decisions. Routing cost is taken into account in an approximate fashion as the product of the number of customers visited in a route and an average cost of travel between customers. The reason for this modeling approximation is that our model is a strategic one integrating several cost factors and routing cost is only one of these factors. Therefore, it is not worth to make the model very complicated to represent only one of the cost components in a detail manner. Besides, the routing cost, in general, does not represent a significant portion in the total cost that we consider in our approach.

In the formulation, we use the notation given below:

Sets:

I : Set of customers

V : Set of vehicles

F : Set of frequencies, F= {1, 2, ..., 21}

F_j : Set of coinciding frequencies, $\forall j = 1, \dots, n$

D : Set of days of the week, D = {1, 2, …, 5}

H : Set of weeks per year, H = {1, 2, … , 52}

Parameters:

n : Number of coinciding frequency sets

m : Number of customers

r_v : Approximate routing cost between two customers (fixed per kilometer cost of each vehicle)

g : Dead heading cost

a_v : Annual ownership cost of vehicle type v

λ_{if} : Annual demand of customer i

h : Annual inventory holding cost per unit of a product

k_{if} : Fixed cost of replenishing customer *i* using frequency *f*

s_{max}: Maximum number of customers that can be visited during the day

c_v : Capacity of vehicle type v

M : A big number

p_f : Total number of annual replenishments for frequency *f*

t_{ik} : Incidence matrix of customers i and k (customers i and k can be in the same route if t_{ik}=2, and cannot be in the same route when t_{ik}=1)

Using the notations above the mathematical model can be represented as follows:

$$Min \ \sum_{v\in V} gp_1 L_{v1} + \sum_{d\in D}\sum_{h\in H}\sum_{v\in V} gR_{dvh} + \sum_{v\in V}\sum_{f\in F} r_v . p_f . C_{vf} + \sum_{v\in V} a_v . V_v + \sum_{i\in I}\sum_{v\in V}\sum_{f\in F} \lambda_{if} . h . X_{ivf} . \left(\frac{1}{2}\right) + \sum_{i\in I}\sum_{v\in V}\sum_{f\in F} k_{if} X_{ivf} \quad (2.1)$$

Subject to:

$$\sum_{i\in I} X_{ivf} = C_{vf} \qquad \forall\, v \in V, \forall f \in F \quad (2.2)$$

$$\sum_{v\in V}\sum_{f\in F} X_{ivf} = 1 \qquad \forall\, i \in I \quad (2.3)$$

$M\ L_{vf} \geq C_{vf}$ $\forall\ v \in V, f \in F$ *(2.4)*

$L_{vf} \leq C_{vf}$ $\forall\ v \in V, f \in F$

$s_{max} V_v\ \geq \sum_i \sum_{f \in F} X_{ivf}$ $\forall\ v \in V$ *(2.5)*

$\sum_{i \in I} \lambda_{if} X_{ivf} \leq c_v$ $\forall v \in V, \forall f \in F$ *(2.6)*

$\sum_{i \in I} \sum_{f \in F_j} \lambda_{if} X_{ivf} \leq c_v$ $\forall\ v \in V, \forall j = 1, \dots, n$ *(2.7)*

$$\sum_{f \in F_j} C_{vf} \leq\ s_{max}$$ $\forall\ v \in V, \forall j = 1, \dots, n$ *(2.8)*

$\sum_{f \in F_j} X_{ivf} + \sum_{f \in F_j} X_{kvf} \leq t_{ik}$ $\forall v \in V, i \in I, k \in I, \forall j = 1, \dots, n$ *(2.9)*

$R_{dvh} \geq L_{vf} - L_{v1}$ $\forall v \in V,\ d \in D, f \in Fj, h \in H$ *(2.10)*

$X_{ivf} \in \{0,1\}$ $\forall\ i \in I, \forall v \in V, \forall f \in F$ *(2.11)*

$L_{vf} \in \{0,1\}$ $\forall v \in V, \forall f \in F$ *(2.12)*

$C_{vf} \in Z_{\geq 0}$ $\forall v \in V, \forall f \in F$ *(2.13)*

$V_v \in \{0,1\}$ $\forall v \in V$ *(2.14)*

$R_{dvh} \in \{0,1\}$ $d \in D, v \in V,\ h \in H$ *(2.15)*

Objective of the problem is to minimize the total cost of transportation, inventory holding and replenishment operations. Transportation costs are presented in the form of separate elements such as dead heading costs, routing costs as an approximate value, and ownership cost. Here in our model, dead heading cost is defined as the cost of travelling from vendor to the first customer and from the last customer in a route back to the vendor. Routing costs are integrated to the model as approximate values. Cost of making a route is given by the multiplication of number of customers

which are replenished by vehicle type v and frequency f and an average cost per kilometer of vehicle type v. Annual ownership cost is calculated as the product of number of type v vehicles and the corresponding ownership cost for that type of vehicle. Other two cost components (holding and replenishment costs) are used as it is in well-known EOQ model.

By constraint (2.2) we determine the number of customers being replenished by vehicle type v and frequency f. Demand satisfaction condition is forced by adding constraint (2.3) to the model. With constraint (2.4), we make sure that L_{vf} is one if we use any vehicle v and frequency f to cover any customer demand. By utilizing (2.5) we determine if a vehicle is used for replenishing any customers, or not.

It is always important not to overload the vehicles. To meet this restriction, we utilized constraint (2.6). Total amount being shipped to customer i which is replenished by vehicle type v and with frequency f must be less than available capacity of vehicle v. This condition must hold for all of the 21 available frequencies. In our problem, the largest loads are shipped on coinciding frequencies and capacity must not be exceeded, and this restriction is forced by constraint (2.7). Obviously, in real life situations, the number of customers which can be visited on a specific route depends on the distances, traffic and some other factors. We embedded this constraint in our model as constraint (2.8), which forces the routes to consist of customers less than a predetermined maximum number of customers. For tackling geographically dispersed customers we used constraint (2.9) which allows us to take some customers in a route and exclude the ones which are not eligible to enter this specific route. Entrance eligibility is given by the incidence matrix, which has value of 1 representing the customers that are not allowed to be on the same route, and value of 2 for the ones which are eligible to enter the same route. By addition of constraint (2.10) we omit the extra repetition of deadheading cost. To handle this extra cost of redundant dead headings we check if there is any vehicle that replenishes any customer with daily frequency. In the case that such a customer exists, all other frequencies assigned to that specific vehicle will be set to zero and their deadheading cost will not affect the objective function value. Actually, in this constraint we subtract L_{v1} from all other L_{vf} variables and set R_{dvH} as greater or

equal to subtraction of these two variables. In this way if for example vehicle 1 replenishes any customer daily, all of the R_{d1H} variables become zero and daily trips number are multiplied with fixed deadheading cost parameter. On the other hand, if no vehicle is assigned to daily frequency, based on the correspondence between f values, d and H (for example f =3 means weekly Tuesdays and corresponds to the second day of the week, that is d = 2 and H = 1,.., 52) the related R_{dvH} is equated to 1 and is multiplied by deadheading cost. (2.11)- (2.15) present the domain for the decision variables.

2.1 Problem Complexity Analysis

In this section we show that the problem tackled in this thesis is a NP-Hard problem through polynomial time reduction from the "One- dimensional bin packing problem" which is proved to be strongly NP hard in E. G. Coffman et al. (1997). In the bin packing problem, objects of different volumes must be packed into a finite number of bins or containers each of volume V in a way that minimizes the number of bins used.

Considering our problem, under some assumptions we can transform the current problem to bin packing problem in pseudo-polynomial time. Consider the scenario for our problem where there is a single vehicle type, a single frequency, no clustering, and all the costs excluding the vehicle ownership cost are equal to zero. Also assume that ownership cost fo a vehicle is equal to 1. This setup is equal to following parameter setting for our problem:

F= {1}, n = r_v = g = k_{if} = h = 0, s_{max}=∞, t_{ik}=2 for all (i, k) pairs, a_v= 1 for all vehicles.

Under this setting our problmes turns into one dimensional bin packing problem, where we try to minimize the number of vehicles.

3. LITRATURE

The problem tackled in this thesis is the problem of replenishment planning along with the fleet size determination. To the best of our knowledge among the researches in the literature, there is not any paper which integrates replenishment and fleet size planning problems. Generally speaking, the researches done on the field of our interest have usually ignored detailed transportation costs which correspond to nearly 60% of total costs in distribution systems. One other aspect which makes our problem different from the researches existing in the literature of subject is the approximate route costs. The works done on routing problems are using vehicle routing solution methods to a great extent which usually causes difficulty in terms of solution times and data gathering. We utilized approximate routing costs in order to simplify the process of problem solution.

Along with all these discrepancies, there are some papers which are close to the problem we considered to some extent. Shipment planning problems which utilize predetermined frequencies, fleet sizing problems, inventory routing problems, the papers which applied fix and optimize heuristic and also the researches about continuous approximation models are among the fields which have some aspects in common with the problem of consideration.

3.1 Predetermined Frequencies

One of the papers which uses predetermined frequencies is Bertazzi et al. (1997). As in our paper, products are shipped from one origin to multiple destinations with a set of given frequencies. Another similarity to our study which makes the mentioned paper interesting for us is the objective of the problem, which is to minimize the costs occurring during transportation and also inventory related costs. Here the approach for solving the problem consists of two phases. First they apply a heuristic based on solving the model as a single link problem. The second phase is dedicated to local improvement of the solution achieved in the previous step. In Bertazzi and

Speranza (2002), the problem of shipping several products from a common origin to a common destination is considered. The objective function of the problem aims at minimizing the sum of inventory and transportation costs as it is in our problem. There are two cases considered in Bertazzi and Speranza (2002): deriving a shipping strategy in existence of continuous frequencies and a set of given discrete frequencies. In this work transportations costs are not considered in detail. That is, dead heading costs and vehicle ownership costs are not integrated to the model. These cost factors constitute a significant part of distribution systems' total costs. Another important point which is neglected in the studies existing in the litrature is that generally routing costs are not considered. Another outsanding charactristic of our problem which makes it significant is that all the inventory related costs, i.e. holding cost, replenishment costs are considered in our model.

Another work considering feasible predetermined frequencies is Maria Grazia Speranza and Ukovich (1994). In this work, integer and mixed integer linear programming models are developed for four situations. Assumptions of the problem tackled in this study are the proportionality of transportation costs with the number of journeys that a typical vehicle makes, and the demand divisibility. Luca Bertazzi (2000) considers the problem of shipping several products from an origin to a single destination under a predetermined frequency set. In this paper dominance rules are derived and the efficiency of a branch and bound algorithm is improved using these dominancy rules. Additionally, some heuristics are suggested and compared with EOQ-type algorithms. Bertazzi and Grazia Speranza (1999) investigates the problem of minimizing sum of the inventory and transportation costs in the multi-products logistics with one origin, some intermediate nodes, and a destination in existence of predetermined frequency set. As a solution approach heuristics based on decomposition of sequences or based on the solution of simpler problems are proposed. One other research work done by Bertazzi et al. (2005), which also has some aspects in common with the subject under consideration in this paper, tackles a complicated production-distribution system in which several items are produced repeatedly, and they are distributed to a set of retailers using a fleet of vehicles. VMI strategy studied in their work is evaluated in two categories and two different decompositions of the problem along with presented optimal or heuristic procedures

for the solution of the sub problems. In M. G. Speranza and Ukovich (1996) again the problem of distributing various products from a single source to a multiple set of destinations in existence of the given frequencies is studied. Here, the Np-hardness of the problem is proved and a branch and bound method is applied as a solution method.

In general, in works which have considered predetermined frequencies the replenishment costs are not taken into consideration. Additionally, ownership costs are ignored and are not included in the mathematical models which they suggested. Another difference of our model from these works is that we consider routing costs in two different parts, i.e. deadheading and cost of travelling between two customers, while in predetermined frequency works transportation costs are taken as a value proportional to number of trips that a vehicle makes. In our model, we consider coinciding frequency effects which is neglected in the works mentioned above. To summarize, our problem is more detailed and includes all the costs which can effect the replenishment and fleet sizing decisions.

3.2 Fleet Sizing Problems

In Baldacci et al. (2008) an overview of approaches for solving heterogenous VRPs is given and as no exact algorithms has been presented for the problem under consideration, some lower bound assessments are given. In Żak et al. (2011) a fleet sizing problem in a road freight transportation company with heterogeneous fleet is considered. Sayarshad and Ghoseiri (2009) suggested a new formulation and solution procedure for optimizing the fleet size and freight car allocation wherein as in our problem car demands are assumed to be deterministic. Crainic (2000) introduces a new classification of service network design problems and formulations in addition to presentation of a state-of-the-art review about studies on service network design modelling and mathematical formulation developments for network design. Another research paper which deals with a problem that has some aspects seeming similar to the problem we have focused on in here, is the work done by Desrochers and Verhoog (1991). In their paper, a problem of simaltaneously selected composition and routing of a fleet of vehicle is addressed wherein customer demands are known and are from a centeral depot. As a solution approach, a new savings heuristic is

used. One of the aspects which makes our problem different from their work is that we utilize a strategic routing problem. To illustrate, fleet sizing problems in general use exact routing algorithms which requires large amounts of precise data and excessive solution times. Conversely, in our research we use average cost of the leg between two customers as an approximate value which will simplify the problem solution to a large extent considering the slight effect that routing cost has in total transportation costs.

3.3 Inventory Routing Problems

The inventory routing problem is a problem in which inventory management, vehicle routing and delivery schedualing decisions are made simaltaneously. Generallay, in these class of optimization problems, a single product is shipped from a single origin to multiple destinations in a period of T while total cost of all operations is minimized. The demand for a typical customer i is equal to u_i and each customer is able to keep a local inventory of product up to a maxmimum of C_i. Customer i has the inventory equal to I_i at time 0. For accomplishing shipments a homogeneous fleet of m vehicles is available. Carrying capacity of each vehicle which is included in the available fleet is equal to Q. Here the objective is to minimize the distribution costs and also obstruct stockouts at any of the customers.

According to Campbell et al. (1998) in a typical inventory routing problem we are about to make three important decisions as below:

- When to serve a customer?
- How much to deliver to a customer when it is served?
- Which delivery routes to use?

One of the papers which we analyzed under the category of IRP(Inventory Routing Problem), is Leandro C. Coelho (2014). In this research inventory routing problem is defined as a combination of vehicle routing and inventory management problems in which a supplier has to deliver products to a number of geographically dispersed customers, subject to side constraints. The paper aims at reviewing the IRPs with respect to their structural variants and the availability of information in customer demand. Coelho and Laporte (2013) has proposed a branch and cut algorithm for the

exact solution of several categories of inventory routing problems. Another review paper is the one by Moin and Salhi (2007) which classifies the models of IRP based on the planning horizon they have employed. The aspect which makes IRPs similar to problem of our interest is that in both problems demands of geographically dispersed customers are distributed.

The most important differences of our problem from IRPs are the main focuses of our problem, which are determining the composition of the fleet considering the ownership cost, and the assignment of both frequency and vehicles to customers considering detailed inventory related costs. In terms of delivery frequencies, as mention in problem definition section, we utilize predetermined frequencies which, to the best of our knowledge, is not generally utilized in inventory routing problems. Inventory routing problems are extensions of vehicle routing problems and are classified as operational problems. Here, we do not attack an operational problem, instead we suggest an strategic problem which uses approximate routing costs instead of solving vehicle routing problems.

3.4 Fix And Optimize Heuristic

In order to understand the heuristic method which we decided to apply to our model in this thesis research comprehensively we investigated the papers which applied this approach to various problems previously. One of the researches is Gintner et al. (2005) in which fix and optimize heuristic is used for bus scheduling problem. Federgruen et al. (2007) has applied the fix and optimize heuristic to multi-products, capacitated lot sizing problem. In Helber and Sahling (2010) the same heuristic is utilized for soling the multi-level capacitated lot sizing problem. One problem which has some similarities to our problem is the one studied in Dorneles et al. (2014). The problem considered in this research is a full integer problem with all variables defined as binaries except for one integer variable. Fix and optimize heuristic is applied to the problem and the efficiency of the algorithm is analyzed. It is stated that the proposed fix and optimize heuristic were able to find new best known solutions for seven instances including three optimal ones.

3.5 Continuous Approximation Models

As mentioned before, here we consider a strategic problem, thus the routing costs are taken as approximated values. Jabali et al. (2012) presents a cotinuous approximation model for determining the long-term vehicle fleet composition needed for performing distribution activities. As in our problem, vehicles differ in terms of their capacities, fixed cost per kilometer and an extra difference which is route durations. The assumption here is that customers are dispersed in a circular service region which is partitioned into zones that each of them are serviced by a single vehicle. The routing costs are assessed by means of a continous approximation model as we will do in our research. Huang et al. (2013) also uses continuous approximation model for routing teams to different communities to assess damage and relief needs following a disaster. The model named as continuous approximation model yields solutions which are easily implemented and reduces the neccassity for detailed data and computational requirements. In our problem we use the routing cost of a leg between two customers as an approximate value. As mentioned before, in total, routing costs constitute a small portion of costs of distribution systems in comparison with total cost of a product. As a result, handling the routing costs as approximate values will not affect the decisions which are to be made in a great extent inspite the simplification it causes in problem solution. In the problem which we have worked on, we calculate the routing costs as multiplication of an approximate route cost, number of customers being replenished with a specific vehicle type and a specific frequency and total number of replenishments per year.

4. NUMERICAL ANALYSIS OF THE PROBLEM

The problem tackled here, is an integration of fleet sizing and replenishment planning problem and it is presented as a mixed integer programming model. Several customers must be replenished by a single frequency and a single vehicle. In order to have an understanding of the behavior of the solution of our problem and derive some managerial insight, we have solved the model under four different demand scenarios, and 24 different parameter settings. We used CPLEX OPL V. 12.4 on a PC with Intel Core i7-3612QM 2.10 GHz processor, 6.00GB of RAM. Solution time for all of the scenarios was under 2 hours except for the parameter settings which were not solved to the optimal in 2 hours. The problems solved optimally took a time period from a few minutes to more than half hour.

In this section, we will first introduce the data set we utilized, then we will explain the scenarios and parameter settings. Finally will present analysis based on the results we obtained.

4.1 Data Set

We worked on a problem composed of 20 customers with deterministic demand. The annual demand values are sampled randomly from a uniform distribution between 80 and 120 ($D_i \in$ Uniform [80,120]). The demand is presented on annual basis considering the possible frequencies. We have a given set of discrete frequencies, which are daily, weekly, biweekly, thrice-weekly, and quarto-weekly. Thus, we have 21 different frequencies available (5*4+1). As mentioned previously, demand has been adjusted considering available frequencies, that is, the annual demand for a customer being replenished every week is calculated as “annual demand/number of weeks in a year “, etc. Another cost parameter to be considered is the cost of replenishment operations presented by “k”. k is also adjusted according to the available set of frequencies, that is, the k value for a customer being replenished biweekly is calculated as “k *(number of weeks in a year/2)”. Here k is equated to

50. Since replenishment costs are calculated as multiplication of k, annual replenishment number and the customer demands, number of possible replenishment in one year must be applied to the model. For this purpose, we defined parameter p_f which represents the number of replenishments during one year under frequency f. To illustrate more, suppose one customer is replenished thrice-weekly. The number of replenishments occurring during a year of thrice-weekly replenishments is calculated as: P(f) = 52/3 (It is obvious that we have 52 weeks in a year). Holding costs usually play an important role in making replenishment decisions. Sometimes extremely high holding cost factors force the decision maker to choose more frequent replenishment plans in order to trade-off the replenishment and holding costs. Conversely, a system with very low holding costs will prefer less frequent replenishments to save more of transportation and replenishment costs. Thus, holding cost is an important factor, here in our study, we used a holding cost of 300 and 600 for different scenarios.

For accomplishing transportation operations, we have unlimited number of heterogeneous vehicles. Mainly, the vehicles differ in carrying capacity, ownership cost and costs per kilometer. In terms of carrying capacity, we have three types of vehicles in hand. Capacity of these vehicles is calculated based on weekly, bi-weekly and thrice-weekly demands of customers. Average annual demand of customers is equal to 100. On average, the daily demand of a customer is about 0.27397 (100/365). Calculating the same customer's weekly demand we have "number of days in a week * daily demand" which is equal to: 5*0.27397=1.36986. As mentioned before, we have a constraint on the maximum number of customers which a vehicle can visit on a single day. Each vehicle can at most visit 5 customers during a single day and thus the lot size which it carries is equal to 5*1.36986= 6.85~7. Thus, the smaller vehicle's carrying capacity is set to 7 units. The same procedure was applied to the vehicles with carrying capacity based on bi-weekly and thrice-weekly replenishment plans and the result was 14 and 21 units. Carrying capacity for the larger vehicles is calculated as "smaller capacity+40%*smaller capacity" which equates to 10 units for smaller vehicles with capacity 7, to 20 for smaller vehicles with capacity 14 and to 27 for smaller vehicles with capacity 21.

Ownership costs include all the vehicle related costs such as depreciation costs, labor costs, taxes and etc. The ownership cost for smaller capacities is equal to 40800 TL/year. Ownership costs were adopted from Ertogral and Gonzalez (2015), and reflect real life situation costs. For the case in which larger carrying capacities are considered, two situations are tackled. One of this situations is the one in which the vehicle with larger capacity cost the vendor "40800+ 20%*40800" and the other situation is which the vendor faces the cost equal to "40800+ 40%*40800".

In addition to ownership costs, cost per kilometer factor affects the total expenses. In general, among the papers existing in the literature of inventory routing and distribution systems, routings is done based on solving normal VRP or IRPs. Bringing optimal or near optimal solutions to routing problems needs accurate data and also a plentiful amount of solution times. This aspect of routing problems makes them difficult for handling in everyday life. Here in this thesis, we used cost per kilometers as approximate values which will simplify the process of solving the problem. Cost per kilometer of vehicles, r_v, with smaller carrying capacity is equated to 7 which was adopted from the research done by Ertogral and Gonzalez (2015). Cost per kilometer of vehicles with higher carrying capacity is calculated as "cost per kilometer of smaller vehicles + 20% *cost per kilometer of smaller vehicles" and "cost per kilometer of smaller vehicles + 40% *cost per kilometer of smaller vehicles ".

In some real life situations we face the customers that are dispersed widely in very distant geographical locations. Putting all these customers together on the same route will not be possible all the time. To handle such circumstances, we defined an incidence matrix which represents the fact that customer i cannot be on the same route with customer k or vice versa. Elements of incidence matrix are expressed as value of 1, standing for the customers which are not possible to be visited on the same routes, and 2 for the ones who are free to be on the same routes.

In order to evaluate the results under various scenarios we made use of the incidence matrix in which the customers with the possibility of allocation to the same routes are shown with number 2, and the customers which cannot be on the same route are presented by 1. One other parameter used in the model, is H which stands for the

frequencies on the basis of weeks and it is used for deletion of the redundant repetitions occurring on the coinciding frequencies. To illustrate more, H for weekly replenished customers is presented as 1,2,3,…,52 whereas in case of customers being replenished thrice-weekly have H values of 3,9,..,52.

Scenarios being considered here are the problems with normal demand and no/four clusters, and with 50% higher demand no/ four clusters. In each of these four scenarios, effects of each factor (holding cost, cost per kilometer, capacities and…) are investigated and also the changes appeared in costs in cases of the various scenarios are analyzed. The scenarios and parameter settings are as shown below in Table 4.1 and Table 4.2.

Table 4.1: Demand Scenarios and their indicators.

Scenarios	**Indicators**
No cluster, Normal demand	1
Clustered, Normal demand	2
No cluster, 50% increased demand	3
Clustered, 50% increased demand	4

Table 4.2: Parameter settings and indicators.

Parameter Setting					Indicator
k=50	h=300	A 20%	R20%	Cap 7,10	1
				Cap 14,20	2
				Cap 21,27	3
			R40%	Cap 7,10	4
				Cap 14,20	5
				Cap 21,27	6
		A 40%	R20%	Cap 7,10	7
				Cap 14,20	8
				Cap 21,27	9
			R40%	Cap 7,10	10
				Cap 14,20	11
				Cap 21,27	12
	h=600	A 20%	R20%	Cap 7,10	13
				Cap 14,20	14
				Cap 21,27	15
			R40%	Cap 7,10	16
				Cap 14,20	17
				Cap 21,27	18
		A 40%	R20%	Cap 7,10	19
				Cap 14,20	20
				Cap 21,27	21
			R40%	Cap 7,10	22
				Cap 14,20	23
				Cap 21,27	24

4.2 Results And Analysis

In this section, we would like to investigate the way that clustering and increase in demand affect the results we get from the problem. In Chart 4.1 we represent the changes in the objective function value in percentages in comparison to base case of each scenario (no cluster, normal demand case is assumed as base for normal demand scenarios, and high demand no cluster is the basis for other two cases).

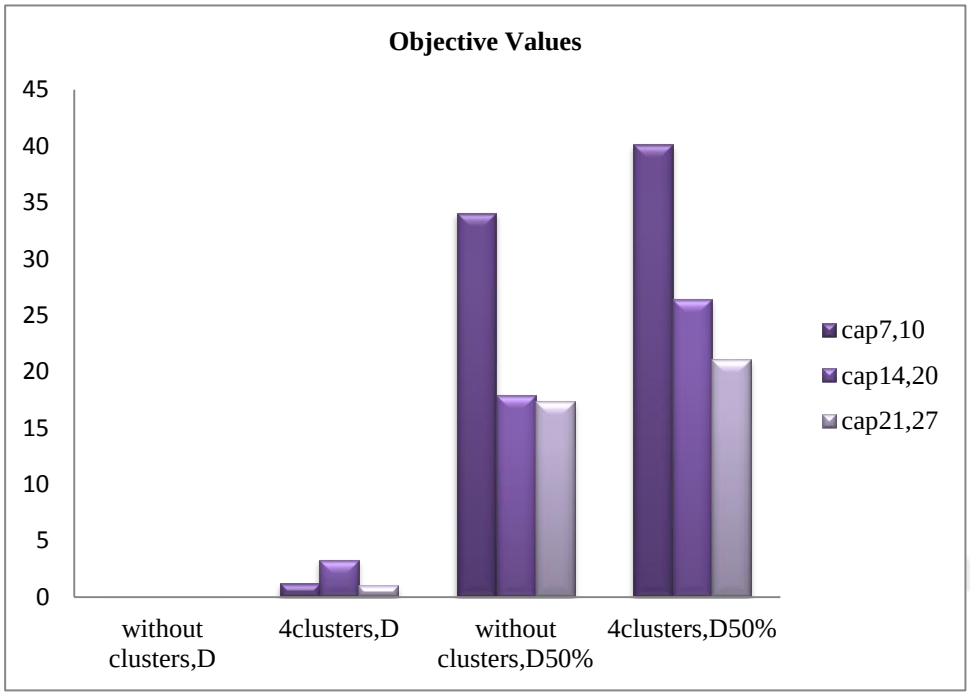

Chart 4.1: Percentage changes of objective function values.

As it is obvious from the chart, the addition of clusters and increase in demand result in an increase in the objective function value. This increase is sharper for the problems with vehicle capacities of (7,10). This may be due to the capacity limitation which will lead to higher routing costs or replenishment costs. Lower capacities limit the size of the lots which are transported and this will necessitate more frequent replenishments. One of the important parts of objective function is routing cost. In our model, routing cost consists of dead heading costs and travel costs between customers.

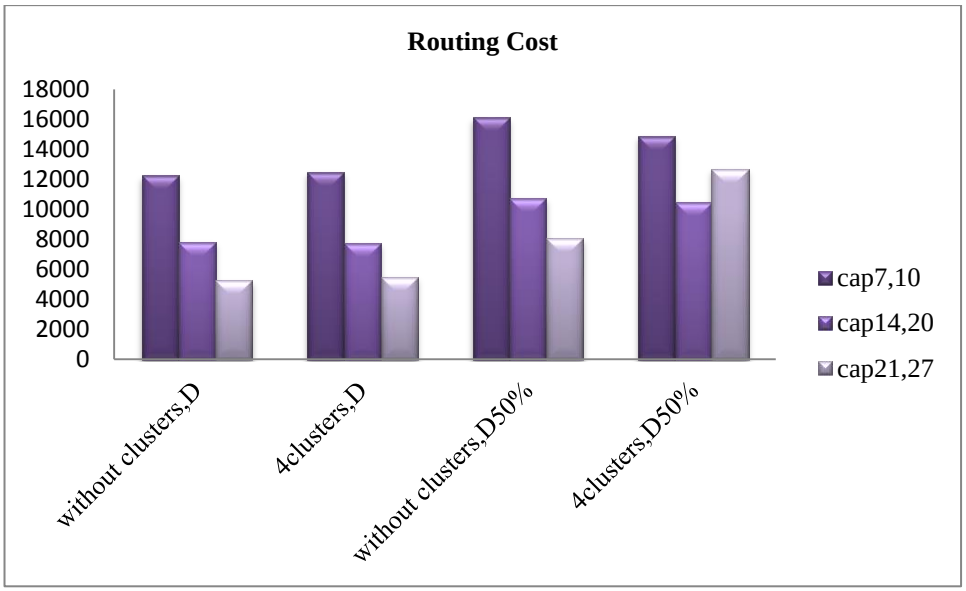

Chart 4.2: Percentage changes of routing costs

Routing costs decrease proportional to increase in carrying capacity. The reason is that the lot consolidation possibility is proportional to vehicle capacity. The higher vehicle capacities facilitate more lot consolidation. As it is revealed in the chart, in scenario 4 routing costs increases when the vehicle capacity increases from 14 and 20 to 21 and 27. This increase is due to the utilization of smaller vehicles instead of a combination of large and small vehicles. This change of vehicle utilization may cause the number of replenishments to increase which results in higher deadheading costs. Another situation in which routing costs escalate is demand increase. The reason is that satisfying increased demands with the same vehicle capacity yields utilization of more vehicles which imposes the extra ownership costs to routing cost.

Ownership costs also reveal changes along different scenarios.

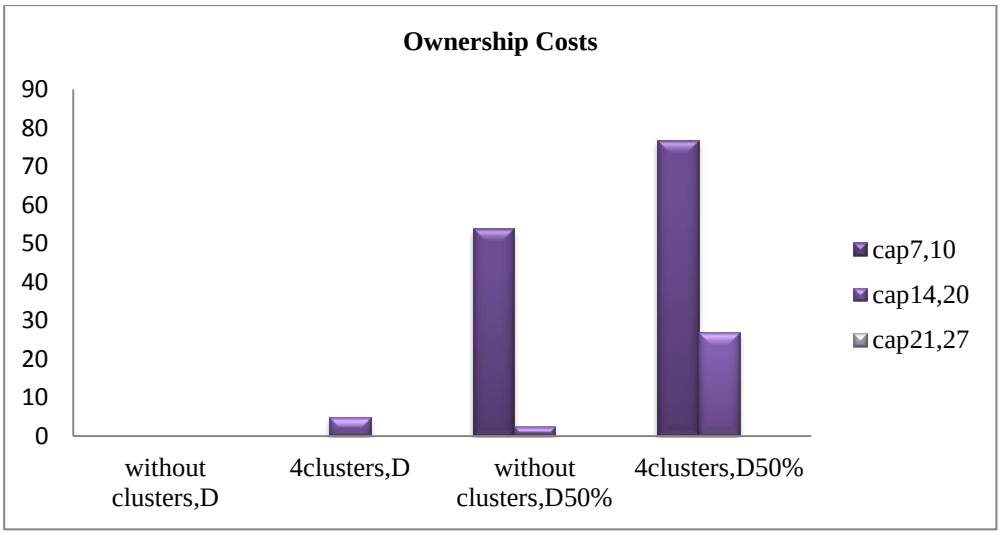

Chart 4.3: Percentage changes of ownership costs

The addition of clusters, does not affect the utilization of vehicles with capacities 27 and 21 units. Identically, demand increase does not change the number of vehicles which have capacities equal to 21 and 27 units. That is, the same number of vehicles with capacity 21 is used along all of the scenarios. This may be due to the relation of capacity and lot sizes. The capacity of 21 units is satisfactory amount for handling replenishment of customers with these demand values.

The ownership cost for vehicles with capacities of 7 and 10 increases when customers are divided into clusters and demand is increased. This increase in ownership costs is due to low possibility of lot consolidation in clustered case which leads to utilization of higher number of vehicles. In addition to clustering, increase in demand causes the lot sizes to become larger and it forces the usage of more vehicles to compensate the disadvantage of low capacity. The ownership cost for vehicles with capacities 14 and 20 increases by clustering the customers. The reason for this increase is that optimization tries to reduce the replenishment cost and also routing cost by means of using maximum possible capacity that allows lot consolidation.

Ownership cost for the scenario with increased demand and clusters is less than the cost for normal demand and clustered case. The reason for this situation can be explained in terms of vehicle type utilization. In the case with clusters vehicle capacity is used in a way that affords maximum consolidation thus a combination of

high and low capacities are used. Conversely, in the case with increased demand and no clusters, consolidation is allowed and no restrictions exist except the vehicle capacity. Thus, there must be a trade-off between ownership and replenishments costs. This tradeoff leads into using low capacity vehicles.

The reason for low capacity utilization is that they have enough capacity which means that the difference between 20 and 14 units of capacities does not worth paying the difference in the ownership costs. Thus here the low capacities are preferred which reduce the ownership cost in comparison to the scenario 1.

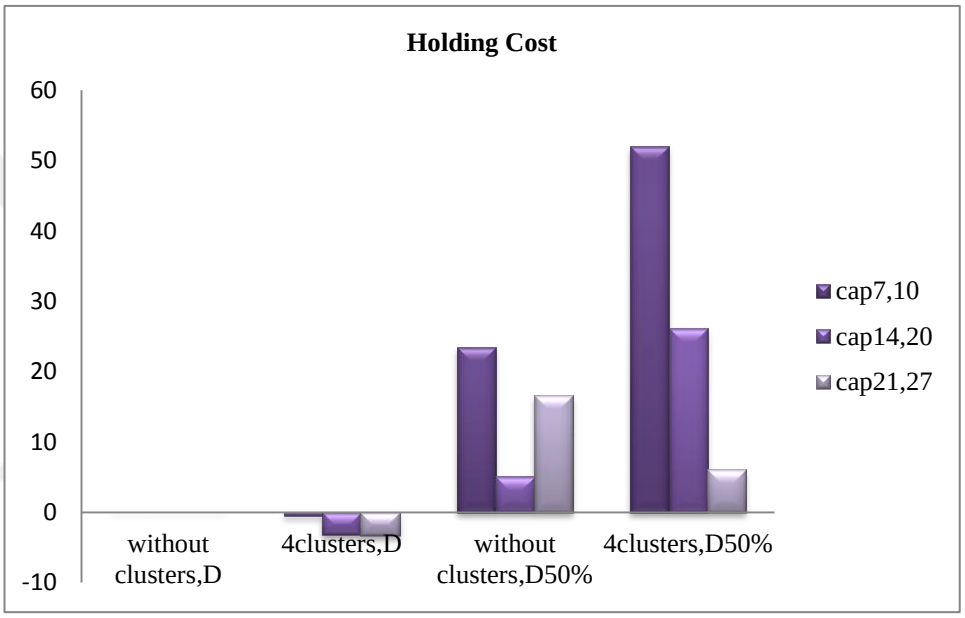

Chart 4.4: Percentage change of holding costs

Holding cost in clustered normal demand scenario decreases for all of the capacities because there is a restriction of lot consolidation resulting from routing rule. Routing rules forces some customers not to be on the same route with some other customers. This means that simply a vehicle cannot consolidate the demand of different customers on the same trip. Another restriction for consolidation is the capacity of the vehicles which do not allow the vendor to send lots larger than a specific size. Smaller lots lead to less holding costs.

Holding costs for vehicles with capacities 7, 10 and 14, 20 increases with the demand increase which may be due to the increase in vehicle utilization. That is, in the

scenario with normal demand and four clusters, total number of vehicles used is equal to 8 while in other scenarios total vehicle number changes to 16 for capacities 7, 10 and 9 for capacities 14, 20.

For the increased demand with clusters the number of vehicles is the same as number of the vehicles in no cluster increased demand case but the combination differs. In this scenario more large vehicle is used which means that much larger lots can be transported. This will lead to higher stocks in warehouses and respectively higher holding costs.

The holding cost value for vehicles with capacities 21 and 27 decreases when customers are divided to clusters and have increased demands. In this case there exists enough capacity for possible consolidation but routing rules will not allow consolidation more than a limited value. Thus the vehicles are transporting lots with lower sizes which cause lower holding costs.

Another cost factor which is affected by the addition of clusters is the replenishment cost.

Chart 4.5 shows how replenishment costs changes.

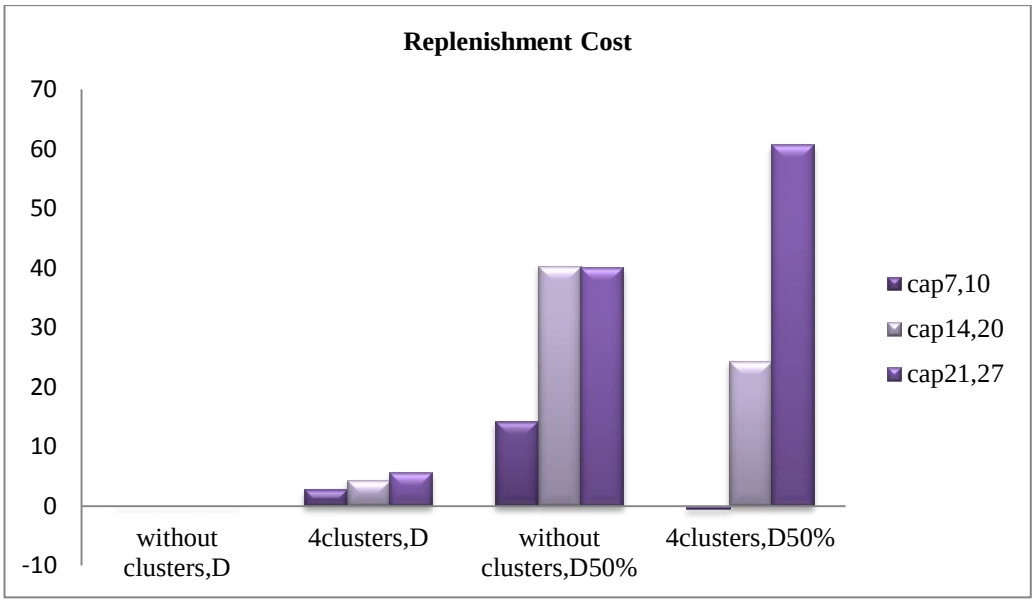

Chart 4.5: Percentage changes of replenishment costs

When clusters are considered in normal demand scenarios, the size of transported lots decreases due to routing rules and thus more frequent replenishments are needed which lead in to higher replenishment costs.

The same pattern is followed by vehicles with capacities 14, 20 and 21, 27 through scenarios 3 and 4. In scenario 4 the replenishment cost for vehicles with capacities 7, 10 decreases. The reason is that an equal number of high and low capacity vehicles are used that have enough capacity for consolidating the amount allowed by routing rules.

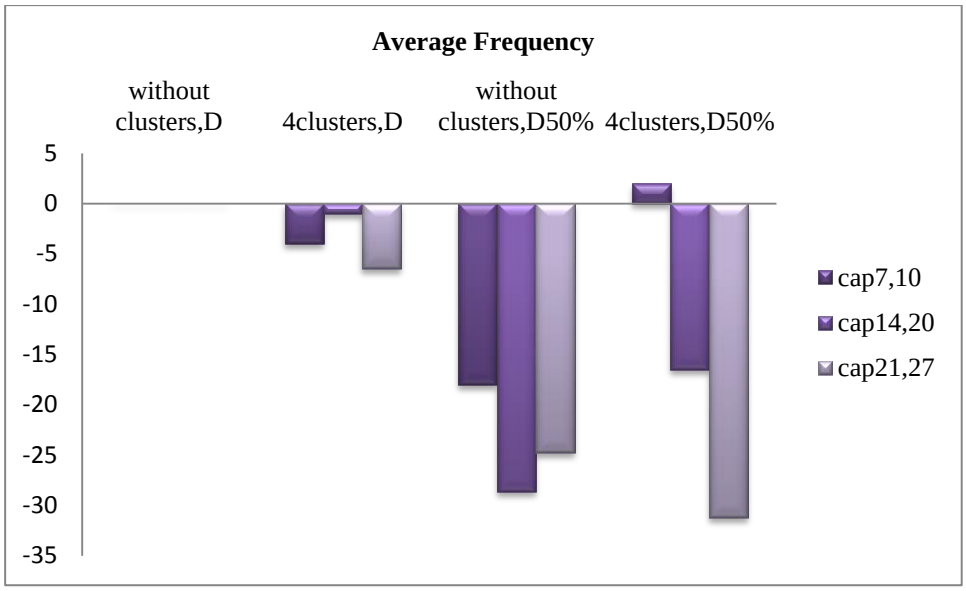

Chart 4.6: Percentage changes of average frequency

Here in this chart, average frequency is calculated as [$(\sum x_{ivf} * \mathrm{P(f)}$)/ number of customers] and the frequencies are numbers like 1.2 (meaning replenishments are done weekly or biweekly), 2.4 (replenishments are done biweekly or thrice weekly). Our expectation about the decrease in frequencies and more frequent replenishments is met in normal demanded scenarios. These decreased frequencies are due to the addition of clusters and decreased consolidation possibility. In the high demand scenarios addition of clusters results is less frequent replenishments. The reason for frequency increase is that by increasing the time between two successive replenishments we have better consolidated oreders. Higher consolidation causes decrease in replenishment costs while increasing the holding cost.

Lot sizes changes is parallel to the changes occurred in average frequencies. In scenarios with normal demand and clusters, both frequency and lot sizes decrease. On the other hand, in the high demand scenarios, higher frequencies cause lot sizes to grow larger. In this case lot sizes increase for capacities equal to 7-10 and 14-20. The reason is that the demand is increased without changing the carrying capacity. Aditionally, in the scenarios with higher demands for the mentioned capacities(7-10, 14-20) the combination of low and high capacity vehicle utilization is different from the scenarios with low demands, i.e, in general higher number of vehicles are utilized and a shift from high capacity utilization to low capacity utilization is observed.

Conversely, for the vehicles with capacities 21-27 with demand increase lot sizes decrease. This change is due to more frequent replenishments which leads to smaller lot transportation.The percentage changes are shown in Chart 4.7.

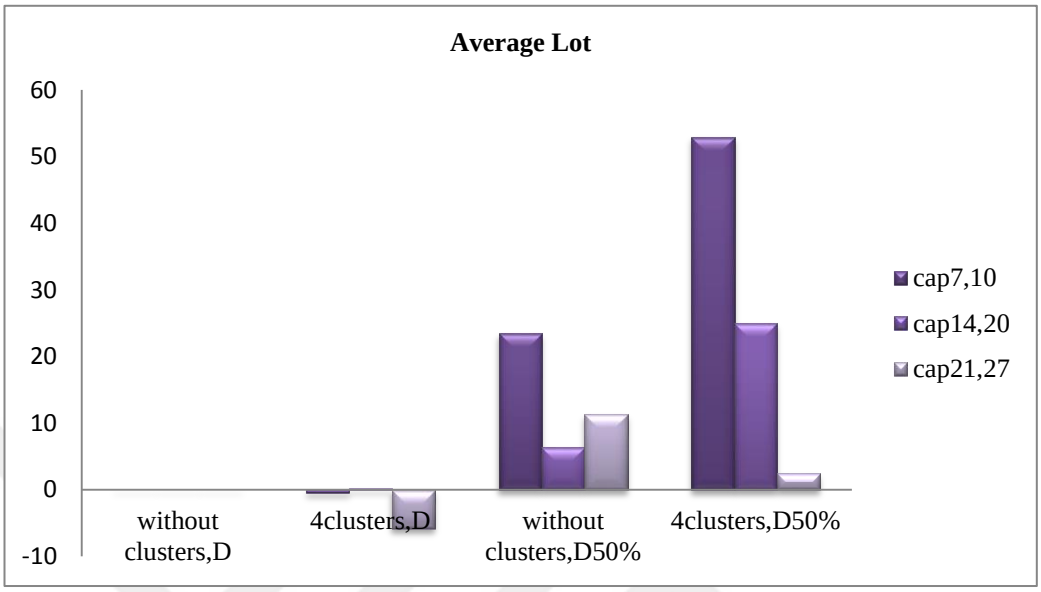

Chart 4.7: Percentage changes of average lot

After analyzing the changes in average lot size, the changes in number of vehicles used in each case should be considered. Generally speaking, the addition of clusters in both cases (high and normal capacities) causes a shif to the large vehicle usage. This happens in order to create a flexibility for demand consolidation. As discussed previously, adding clusters causes less frequent replenishments and larger lot sizes. In order to consolidate demands into these large lot sizes and also for neutralizing the clustering restrictions vehicle selection shifts to large vehicles. The changes are represented in Chart 4.8.

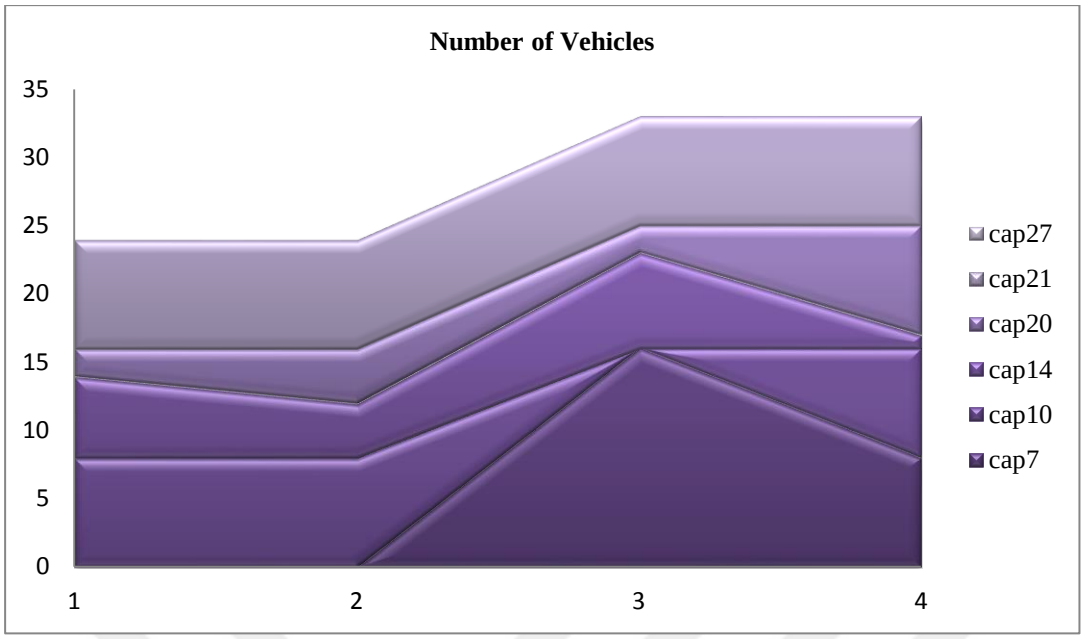

Chart 4.8: Number of vehicles used under each scenario

For problems with vehicle capacities equal to 7 and 10 units, in the first two scenarios using larger capacity vehicles (10 unit) is preferred. This may be for reducing the replenishment costs and also the part of routing costs which is proportional to the repetitions in a year. Demand increase leads to utilization of small capacity vehicles. This will increase the average lots and thus holding costs will go higher. In the case of increased demand the reason for small capacity selection can be the limitation that routing rules create. Using larger number of low capacity vehicles allows to consolidate the lots as much as possible while the routing and ownership costs does not increase as much as they would if we utilized high capacities.

Considering the vehicles with capacities 14 and 20, we see that in scenarios 1 and 3 vehicles with capacity of 14 units are used. This capacity maybe enough for consolidating the lots to a desirable level. In scenario 4, larger demands and low consolidation possibilities lead to utilization of more vehicles of capacity 20 which will consolidate more lots if possible.

In case of capacities 21 and 27, the obvious point is that the capacity of 21 units is clearly enough to handle distribution plans and the lot transportation as capacity 27 is never used in any scenarios.

routes. The reason of this decrease is obviously more consolidated transportation under clustered demand. The changes in route numbers are parallel with our expectations as shown in Chart 4.9.

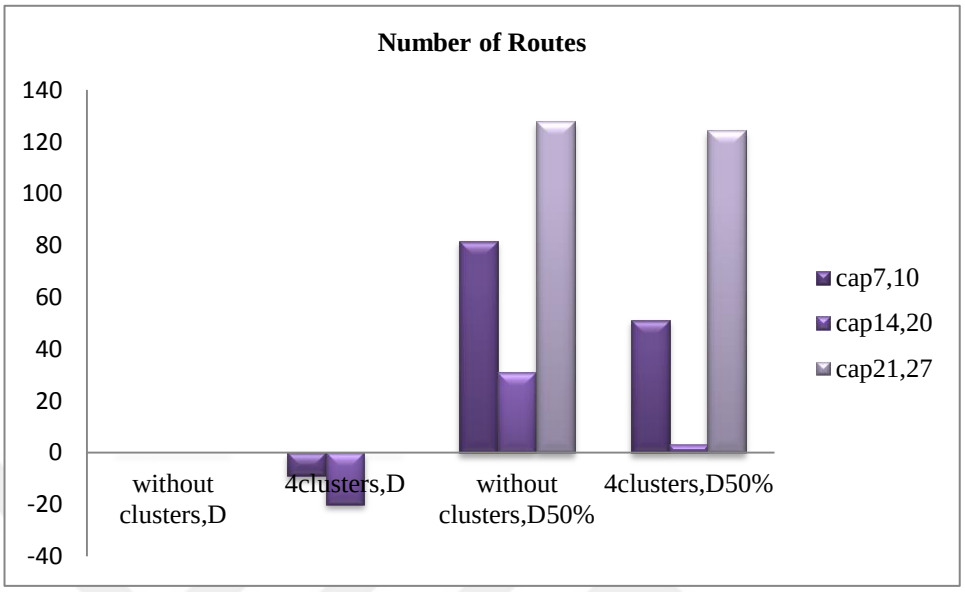

Chart 4.9: Percentage changes in number of routes under scenarios

5. VALID INEQUALITIES AND LOWER BOUND ANALYSIS

5.1 LP Based Lower Bound Analysis

As mentioned previously, special case of the problem studied in this thesis can be shown to be a "Bin Packing Problem", which is known to be Np-hard. This characteristic leads into excessive computational time as the problem size gets larger. For large problems, the performance of our heuristic methods applied to the problem should be compared with the lower bounds of the problems under consideration.

The problem considered in this thesis is in form of an integer programming model and all of the decision variables are binaries or integers. We used two types of relaxations for obtaining lower bounds. First method we used is full LP relaxation, in which all of the binary variables are relaxed in the interval [0, 1] and integer variables are taken as belonging to R^+. For further analysis of bounds we investigated the bounds from partial LP relaxation problems. In this case, all the binary variables except V_v are relaxed in [0, 1] interval and V_v is still defined as a binary variable. The integer variable C_{vf} is set to be in R^+ . As expected, the results with a binary variable and other variables as continuous values show a tighter lower bound for the scenarios under study. Problems were solved with commercial solver CPLEX 12.4. Results for different scenarios and parameter settings are given in tables Table 5.1throughTable 5.4.

Table 5.1: Full and partial LP bounds for scenario 1.

Parameter settings	**Lower Bounds Full LP**	**Lower Bounds Partial LP**	**Optimal**	**Gap From LB Full LP**	**Gap From LB Partial LP**
1	80738.42	110566.40	113304.72	28.74%	2.42%
2	77581.18	88761.72	91365.27	15.09%	2.85%
3	77581.18	80615.21	81868.76	5.24%	1.53%
4	80923.66	111804.10	114578.72	29.37%	2.42%
5	77581.18	88761.72	91503.29	15.21%	3.00%
6	77581.18	80615.21	81868.76	5.24%	1.53%
7	83114.85	118726.40	121464.72	31.57%	2.25%
8	77581.18	88761.72	91503.29	15.21%	3.00%
9	77581.18	80615.21	81868.76	5.24%	1.53%
10	83234.52	119964.10	122738.72	32.19%	2.26%
11	77581.18	88761.72	91503.29	15.21%	3.00%
12	77581.18	80615.21	81868.76	5.24%	1.53%
13	94893.63	118066.40	120527.44	21.27%	2.04%
14	93859.04	99261.72	101778.94	7.78%	2.47%
15	93859.04	95801.90	95981.98	2.21%	0.19%
16	94966.82	119304.10	121801.44	22.03%	2.05%
17	93859.04	99261.72	101778.94	7.78%	2.47%
18	93859.04	95801.90	95981.98	2.21%	0.19%
19	95674.93	126226.40	128687.44	25.65%	1.91%
20	93859.04	99261.72	101778.94	7.78%	2.47%
21	93859.04	95801.90	95981.98	2.21%	0.19%
22	95784.12	127646.10	129961.44	26.30%	1.78%
23	93859.04	99261.72	101778.94	7.78%	2.47%
24	93859.04	95801.90	95981.98	2.21%	0.19%
Average Gap				14.12%	1.91%

Table 5.2: Full and partial LP bounds for scenario 2.

Parameter settings	**Lower Bounds Full LP**	**Lower Bounds Partial LP**	**Optimal**	**Gap From LB Full LP**	**Gap From LB Partial LP**
1	80738.42	110566.37	114788.66	29.66%	3.68%
2	77851.18	88761.72	92188.28	15.55%	3.72%
3	77851.18	80615.21	83268.44	6.51%	3.19%
4	80923.57	111804.09	116099.06	30.30%	3.70%
5	77851.18	88761.72	92846.28	16.15%	4.40%
6	77851.18	80615.21	83268.44	6.51%	3.19%
7	83114.85	118726.37	122948.66	32.40%	3.43%
8	77581.18	88761.72	95850.94	19.06%	7.40%
9	77581.18	80615.21	83268.44	6.83%	3.19%
10	83234.52	119964.09	124259.06	33.02%	3.46%
11	77581.18	88761.72	95850.94	19.06%	7.40%
12	77581.18	80615.21	83268.44	6.83%	3.19%
13	94893.63	118066.37	121976.93	22.20%	3.21%
14	93859.04	99261.72	105218.27	10.80%	5.66%
15	93859.04	95801.90	96341.47	2.58%	0.56%
16	94966.82	119304.09	123287.33	22.97%	3.23%
17	93859.04	99261.72	105489.51	11.03%	5.90%
18	93859.04	95801.90	96341.47	2.58%	0.56%
19	95674.93	126226.37	130136.93	26.48%	3.00%
20	93859.04	99261.72	105489.51	11.03%	5.90%
21	93859.04	95801.90	96341.47	2.58%	0.56%
22	95748.12	127464.09	131447.33	27.16%	3.03%
23	93859.04	99261.72	105489.51	11.03%	5.90%
24	93859.04	95801.90	96341.47	2.58%	0.56%
Average Gap				15.62%	3.67%

Table 5.3: Full and partial LP bounds for scenario 3.

Parameter settings	**Lower Bounds Full LP**	**Lower Bounds Partial LP**	**Optimal**	**Gap From LB Full LP**	**Gap From LB Partial LP**
1	90749.08	147965.46	158635.24	42.79%	6.73%
2	86900.19	104603.93	107246.37	18.97%	2.46%
3	86540.09	94011.72	96680.70	10.49%	2.76%
4	90991.00	147965.46	158635.24	42.64%	6.73%
5	86900.18	105535.89	108192.77	19.68%	2.46%
6	86540.09	94011.72	96680.70	10.49%	2.76%
7	93809.75	147965.46	158635.24	40.86%	6.73%
8	86900.18	107165.46	108993.58	20.27%	1.68%
9	86540.09	94011.72	96680.70	10.49%	2.76%
10	94017.39	147965.46	158635.24	40.73%	6.73%
11	86900.18	107165.46	108993.58	20.27%	1.68%
12	86540.09	94011.72	96680.70	10.49%	2.76%
13	109049.90	158465.46	167550.49	34.92%	5.42%
14	105926.12	117665.46	119415.17	11.30%	1.47%
15	105926.12	109761.72	111975.40	5.40%	1.98%
16	109317.46	158465.46	167550.49	34.76%	5.42%
17	105926.12	117665.46	119415.17	11.30%	1.47%
18	105926.12	109761.72	111975.40	5.40%	1.98%
19	111906.13	158465.46	167550.49	33.21%	5.42%
20	105926.12	117665.46	119415.17	11.30%	1.47%
21	105926.12	109761.72	111975.40	5.40%	1.98%
22	112173.69	158465.46	167550.49	33.05%	5.42%
23	105926.12	117665.46	119415.17	11.30%	1.47%
24	105926.12	109761.72	111975.40	5.40%	1.98%
Average Gap				20.45%	3.40%

Table 5.4: Full and partial LP bounds for scenario 4.

Parameter settings	**Lower Bounds Full LP**	**Lower Bounds Partial LP**	**Optimal**	**Gap From LB Full LP**	**Gap From LB Partial LP**
1	90749.07	147965.46	160457.25	43.44%	7.79%
2	86900.18	104603.93	110814.77	21.58%	5.60%
3	86540.09	94011.72	100685.25	14.05%	6.63%
4	90991.00	147965.46	161405.53	43.63%	8.33%
5	86900.18	105535.89	111884.37	22.33%	5.67%
6	86540.09	94011.72	100685.25	14.05%	6.63%
7	93809.75	147965.46	168593.93	44.36%	12.24%
8	86900.18	107165.46	119128.74	27.05%	10.04%
9	86540.09	94011.72	100685.25	14.05%	6.63%
10	94017.39	147965.46	169565.53	44.55%	12.74%
11	86900.18	107165.46	120088.45	27.64%	10.76%
12	86540.09	94011.72	100685.25	14.05%	6.63%
13	109049.90	158465.46	171362.55	36.36%	7.53%
14	105926.12	117665.46	124261.15	14.76%	5.31%
15	105926.12	109761.72	114577.27	7.55%	4.20%
16	109317.46	158465.46	172345.35	36.57%	8.05%
17	105926.12	117665.46	125353.15	15.50%	6.13%
18	105926.12	109761.72	114577.27	7.55%	4.20%
19	111906.13	158465.46	179522.55	37.66%	11.73%
20	105926.12	117665.46	132421.15	20.01%	11.14%
21	105926.12	109761.72	114577.27	7.55%	4.20%
22	112173.69	15465.46	180505.35	37.86%	91.43%
23	105926.12	117665.46	133513.15	20.66%	11.87%
24	105926.12	109761.72	114577.27	7.55%	4.20%
Average Gap				24.18%	11.24%

As it is presented in the tables, due to full integer nature of the problem under consideration, full LP gaps are relatively high. In our problem the highest costs are related to vehicle ownership which is directly dependent to vehicle utilization. Low full LP bounds is the results of relaxing V_v in [0, 1] interval together with other binary variables. Partial LP bounds performed much better than full LP bounds for nearly all of the scenarios. The highest average gap for partial LP bounds is for scenario4. The improved gaps in partial LP relaxation results is due to keeping vehicle utilization indicating variable as a binary variable.

5.2 Cover Inequalities For Improving Lower Bound

In order to improve the lower bounds from full LP and partial LP relaxations, we generated cover inequalities. According to Wolsey (1998a), a set C ⊆ N is a cover for set X = { x∈ B^n : $\sum_{j=1}^{n} a_j x_j \leq$ b} if $\sum_{j\in C} a_j >$ b. If C ⊆ N is a cover for set X, the cover inequality $\sum_{j\in C} x_j \leq$ |C| -1 is a valid inequality for X. The constraint $\sum_{i\in I} \lambda_{if} X_{ivf} \leq c_v$ which is used to ensure the transported lots is less than each vehicle's capacity is of the type $\sum_{j=1}^{n} a_j x_j \leq$ b with x∈ B^n. We utilized this property to generate for different cover type inequalities: covers using minimum demand, normal cover inequalities, cover inequalities generated by taking alternate demand indices and randomly generated cover inequalities. First type of cover inequalities were generated using Excel functions and the rest were generated using Eclipse Java.

5.2.1 Covers using minimum demand

In covers which are generated using minimum demand of the customers we determined the least demand value among all customers and then found the parameter "W_{vf}" by dividing demand value at each frequency by capacity of each vehicle type. The general idea of this cover type is that the lots carried to customers must be at most as large as vehicle's capacity. Another capacity constraint which must be considered is the capacity constraint in coinciding frequencies. To generate covers for coinciding frequencies, again with the same general idea, we determined the least demand in coinciding frequency sets. All the coinciding frequencies sets contain daily replenishments. Obviously daily demands are the least between other frequency related demands. Therefore, we divided the minimum of daily demands by

the capacity of each vehicle type and named it as "V_{jf}" in which index j represents the index of coinciding frequency sets. The constraints added to model as a result of these operations are as below:

$$\sum_i x_{ivf} \leq W_{vf} \text{ -1} \qquad \forall \text{ v} \in V, f \in F$$

$$\sum_{f \in F_j} \sum_i x_{ivf} \leq V_{vj} \text{ -1} \qquad \forall \text{ v} \in V, j \in n$$

5.2.2 Normal cover inequalities

Utilizing the definitions by Wolsey (1998b) we generated cover inequalities based on demands of the customers using Eclipse Java programming software. We added up the demands of customers progressively until the total demand exceeded the capacity of vehicles which were utilized, i.e. for each carrying capacity a specific set of cover inequalities were generated. At the point in which the total demand is higher than the capacity the total demand is equated to zero and process of demand addition starts from the next customer demand. In this manner we could generate 3 different types cover sets which caused some improvement in lower bounds from full LP relaxation.

5.2.3 Alternate demand cover inequalities

For generating this type of cover inequalities we added up the demands of customers alternately and after each capacity breach the total demand is again equated to zero and the addition process continue starting from the next customer index. As a result of this procedure we could generate 6 different cover sets which are applicable for various carrying capacities.

5.2.4 Random cover inequalities

Random covers were generated using random customer indices. At each iteration we added up the demand of a random customer to the previous random customer's demand and after capacity breach we cleared the total demand and started adding up the demand of other random customers. The important point which must be considered in generating this type of cover inequalities is that covers may have the same customer indices and same total demand values. For handling this problem, at the point in which total demand exceeds the capacity of a specific vehicle, the value

of total demand should be compared to the value of previous total demand array. If the same total does not exist on the previous total demands' array a new cover set will be at hand. As in other processes, the covers are generated for each specific capacity in a distinct way. The results of valid inequality addition are given in Table 5.5 to Table 5.8.

Table 5.5: Lower bounds for scenario 1 after valid inequality addition, Full LP.

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal No Covers**	**Gap From Optimal with Covers**	**Improvement of Gaps**
1	80738.42	80857.03	113304.70	28.74%	28.64%	0.36%
2	77581.18	77581.18	91365.28	15.09%	15.09%	0.00%
3	77581.18	77581.18	81868.77	5.24%	5.24%	0.00%
4	80923.66	81042.90	114578.70	29.37%	29.27%	0.35%
5	77581.18	77581.18	91503.29	15.21%	15.21%	0.00%
6	77581.18	77581.18	81868.77	5.24%	5.24%	0.00%
7	83114.85	83111.85	121464.70	31.57%	31.58%	-0.01%
8	77581.18	77581.18	91503.29	15.21%	15.21%	0.00%
9	77581.18	77581.18	81868.77	5.24%	5.24%	0.00%
10	83234.52	83234.52	122738.70	32.19%	32.19%	0.00%
11	77581.18	77581.18	91503.29	15.21%	15.21%	0.00%
12	77581.18	77581.18	81868.77	5.24%	5.24%	0.00%
13	94893.63	94893.63	120527.40	21.27%	21.27%	0.00%
14	93859.04	93859.04	101778.90	7.78%	7.78%	0.00%
15	93859.04	93859.04	95981.98	2.21%	2.21%	0.00%
16	94966.82	94966.82	121801.40	22.03%	22.03%	0.00%
17	93859.04	93859.04	101778.90	7.78%	7.78%	0.00%
18	93859.04	93859.04	95981.98	2.21%	2.21%	0.00%
19	95674.93	95674.93	128687.40	25.65%	25.65%	0.00%
20	93859.04	93859.04	101778.90	7.78%	7.78%	0.00%
21	93859.04	93859.04	95981.98	2.21%	2.21%	0.00%
22	95784.12	95784.12	129961.40	26.30%	26.30%	0.00%
23	93859.04	93859.04	101778.90	7.78%	7.78%	0.00%
24	93859.04	93859.04	95981.98	2.21%	2.21%	0.00%
Average Gap				14.12%	14.11%	0.03%

Table 5.6: Lower bounds for scenario 2 after valid inequality addition, Full LP.

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal No Cuts**	**Gap From Optimal with Cuts**	**Improvement of Gaps**
1	80738.40	80857.03	114788.70	29.66%	29.56%	0.35%
2	77851.20	77851.18	92188.28	15.55%	15.55%	0.00%
3	77851.20	77851.18	83268.44	6.51%	6.51%	0.00%
4	80923.60	81042.90	116099.10	30.30%	30.20%	0.34%
5	77851.20	77851.18	92846.28	16.15%	16.15%	0.00%
6	77851.20	77851.18	83268.44	6.51%	6.51%	0.00%
7	83114.90	83114.85	122948.70	32.40%	32.40%	0.00%
8	77581.20	77581.18	95850.94	19.06%	19.06%	0.00%
9	77581.20	77581.18	83268.44	6.83%	6.83%	0.00%
10	83234.50	83234.52	124259.10	33.02%	33.02%	0.00%
11	77581.20	77581.18	95850.94	19.06%	19.06%	0.00%
12	77581.20	77581.18	83268.44	6.83%	6.83%	0.00%
13	94893.60	94893.63	121976.90	22.20%	22.20%	0.00%
14	93859.00	93859.04	105218.30	10.80%	10.80%	0.00%
15	93859.00	93859.04	96341.47	2.58%	2.58%	0.00%
16	94966.80	94966.82	123287.30	22.97%	22.97%	0.00%
17	93859.00	93859.04	105489.50	11.03%	11.03%	0.00%
18	93859.00	93859.04	96341.47	2.58%	2.58%	0.00%
19	95674.90	95674.93	130136.90	26.48%	26.48%	0.00%
20	93859.00	93859.04	105489.50	11.03%	11.03%	0.00%
21	93859.00	93859.04	96341.47	2.58%	2.58%	0.00%
22	95748.10	95748.12	131447.30	27.16%	27.16%	0.00%
23	93859.00	93859.04	105489.50	11.03%	11.03%	0.00%
24	93859.00	93859.04	96341.47	2.58%	2.58%	0.00%
Average Gap				15.62%	15.61%	0.03%

Table 5.7: Lower bounds for scenario 3 after valid inequality addition, Full LP.

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal No Covers**	**Gap From Optimal with Covers**	**Improvement of Gaps**
1	90749.10	93859.00	158635.20	42.79%	40.83%	4.58%
2	86900.20	86900.20	107246.40	18.97%	18.97%	0.00%
3	86540.10	86540.10	96680.70	10.49%	10.49%	0.00%
4	90991.00	92044.90	158635.20	42.64%	41.98%	1.56%
5	86900.20	86900.20	108192.80	19.68%	19.68%	0.00%
6	86540.10	86540.10	96680.70	10.49%	10.49%	0.00%
7	93809.80	95661.80	158635.20	40.86%	39.70%	2.86%
8	86900.20	86900.20	108993.60	20.27%	20.27%	0.00%
9	86540.10	86540.10	96680.70	10.49%	10.49%	0.00%
10	94017.40	95930.60	158635.20	40.73%	39.53%	2.96%
11	86900.20	86900.20	108993.60	20.27%	20.27%	0.00%
12	86540.10	86540.10	96680.70	10.49%	10.49%	0.00%
13	109050.00	110176.00	167550.50	34.92%	34.24%	1.92%
14	105926.00	105926.00	119415.20	11.30%	11.30%	0.00%
15	105926.00	105926.00	111975.40	5.40%	5.40%	0.00%
16	109317.00	110511.00	167550.50	34.76%	34.04%	2.05%
17	105926.00	105926.00	119415.20	11.30%	11.30%	0.00%
18	105926.00	105926.00	111975.40	5.40%	5.40%	0.00%
19	111906.00	112414.00	167550.50	33.21%	32.91%	0.91%
20	105926.00	105926.00	119415.20	11.30%	11.30%	0.00%
21	105926.00	105926.00	111975.40	5.40%	5.40%	0.00%
22	112174.00	112605.00	167550.50	33.05%	32.79%	0.78%
23	105926.00	105926.00	119415.20	11.30%	11.30%	0.00%
24	105926.00	105926.00	111975.40	5.40%	5.40%	0.00%
Average Gap				20.45%	20.17%	0.73%

Table 5.8: Lower bounds for scenario 4 after valid inequality addition, Full LP.

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal No Cuts**	**Gap From Optimal with Cuts**	**Improvement of Gaps**
1	90749.10	91776.10	160457.30	43.44%	42.80%	1.47%
2	86900.20	86900.20	110814.80	21.58%	21.58%	0.00%
3	86540.10	86540.10	100685.20	14.05%	14.05%	0.00%
4	90991.00	92044.90	161405.50	43.63%	42.97%	1.50%
5	86900.20	86900.20	111884.40	22.33%	22.33%	0.00%
6	86540.10	86540.10	100685.20	14.05%	14.05%	0.00%
7	93809.80	95661.80	168593.90	44.36%	43.26%	2.48%
8	86900.20	86900.20	119128.70	27.05%	27.05%	0.00%
9	86540.10	86540.10	100685.20	14.05%	14.05%	0.00%
10	94017.40	95930.60	169565.50	44.55%	43.43%	2.53%
11	86900.20	86900.20	120088.40	27.64%	27.64%	0.00%
12	86540.10	86540.10	100685.20	14.05%	14.05%	0.00%
13	109050.00	110176.00	171362.50	36.36%	35.71%	1.81%
14	105926.00	105926.00	124261.10	14.76%	14.76%	0.00%
15	105926.00	105926.00	114577.30	7.55%	7.55%	0.00%
16	109317.00	110511.00	172345.30	36.57%	35.88%	1.89%
17	105926.00	105926.00	125353.10	15.50%	15.50%	0.00%
18	105926.00	105926.00	114577.30	7.55%	7.55%	0.00%
19	111906.00	112414.00	179522.50	37.66%	37.38%	0.75%
20	105926.00	105926.00	132421.10	20.01%	20.01%	0.00%
21	105926.00	105926.00	114577.30	7.55%	7.55%	0.00%
22	112174.00	112605.00	180505.30	37.86%	37.62%	0.63%
23	105926.00	105926.00	133513.10	20.66%	20.66%	0.00%
24	105926.00	105926.00	114577.30	7.55%	7.55%	0.00%
Average Gap				24.18%	23.96%	0.54%

Table 5.9: Lower bounds for scenario 1 after valid inequality addition , Partial LP.

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal**	**Gap From Optimal with Cuts**	**Improvement of Gaps**
1	110566.37	110566.37	113304.70	2.42%	2.42%	0.00%
2	88761.72	88761.72	91365.28	2.85%	2.85%	0.00%
3	80615.21	80615.21	81868.77	1.53%	1.53%	0.00%
4	111804.09	111804.09	114578.70	2.42%	2.42%	0.00%
5	88761.72	88761.72	91503.29	3.00%	3.00%	0.00%
6	80615.21	80615.21	81868.77	1.53%	1.53%	0.00%
7	118726.37	118726.37	121464.70	2.25%	2.25%	0.00%
8	88761.72	88761.72	91503.29	3.00%	3.00%	0.00%
9	80615.21	80615.21	81868.77	1.53%	1.53%	0.00%
10	119964.09	119964.09	122738.70	2.26%	2.26%	0.00%
11	88761.72	88761.72	91503.29	3.00%	3.00%	0.00%
12	80615.21	80615.21	81868.77	1.53%	1.53%	0.00%
13	118066.37	118066.37	120527.40	2.04%	2.04%	0.00%
14	99261.72	99261.72	101778.90	2.47%	2.47%	0.00%
15	95801.90	95801.90	95981.98	0.19%	0.19%	0.00%
16	119304.09	119304.09	121801.40	2.05%	2.05%	0.00%
17	99261.72	99261.72	101778.90	2.47%	2.47%	0.00%
18	95801.90	95801.90	95981.98	0.19%	0.19%	0.00%
19	126226.37	126226.37	128687.40	1.91%	1.91%	0.00%
20	99261.72	99261.72	101778.90	2.47%	2.47%	0.00%
21	95801.90	95801.90	95981.98	0.19%	0.19%	0.00%
22	127646.09	127646.09	129961.40	1.78%	1.78%	0.00%
23	99261.72	99261.72	101778.90	2.47%	2.47%	0.00%
24	95801.90	95801.90	95981.98	0.19%	0.19%	0.00%
Average Gap				1.91%	1.91%	0.00%

Table 5.10: Lower bounds for scenario 2 after valid inequality addition, Partial LP.

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal No Cuts**	**Gap From Optimal with Cuts**	**Improvement of Gaps**
1	110566.37	110566.37	114788.66	3.68%	3.68%	0.00%
2	88761.72	88761.72	92188.28	3.72%	3.72%	0.00%
3	80615.21	80615.21	83268.44	3.19%	3.19%	0.00%
4	111804.09	111804.09	116099.06	3.70%	3.70%	0.00%
5	88761.72	88761.72	92846.28	4.40%	4.40%	0.00%
6	80615.21	80615.21	83268.44	3.19%	3.19%	0.00%
7	118726.37	118726.37	122948.66	3.43%	3.43%	0.00%
8	88761.72	88761.72	95850.94	7.40%	7.40%	0.00%
9	80615.21	80615.21	83268.44	3.19%	3.19%	0.00%
10	119964.09	119964.09	124259.06	3.46%	3.46%	0.00%
11	88761.72	88761.72	95850.94	7.40%	7.40%	0.00%
12	80615.21	80615.21	83268.44	3.19%	3.19%	0.00%
13	118066.37	118066.37	121976.93	3.21%	3.21%	0.00%
14	99261.72	99261.72	105218.27	5.66%	5.66%	0.00%
15	95801.90	95801.90	96341.47	0.56%	0.56%	0.00%
16	119304.09	119304.09	123287.33	3.23%	3.23%	0.00%
17	99261.72	99261.72	105489.51	5.90%	5.90%	0.00%
18	95801.90	95801.90	96341.47	0.56%	0.56%	0.00%
19	126226.37	126226.37	130136.93	3.00%	3.00%	0.00%
20	99261.72	99261.72	105489.51	5.90%	5.90%	0.00%
21	95801.90	95801.90	96341.47	0.56%	0.56%	0.00%
22	127464.09	127464.09	131447.33	3.03%	3.03%	0.00%
23	99261.72	99261.72	105489.51	5.90%	5.90%	0.00%
24	95801.90	95801.90	96341.47	0.56%	0.56%	0.00%
Average Gap				3.67%	3.67%	0.00%

Table 5.11: Lower bounds for scenario 3 after valid inequality addition (Partial LP).

Parameter settings	**Lower Bounds**	**Lower Bounds after Covers**	**Optimal**	**Gap From Optimal No Covers**	**Gap From Optimal with Covers**	**Improvement of Gaps**
1	147965.46	147965.46	158635.20	6.73%	6.73%	0.00%
2	104603.93	104603.93	107246.40	2.46%	2.46%	0.00%
3	94011.72	94011.72	96680.70	2.76%	2.76%	0.00%
4	147965.46	147965.46	158635.20	6.73%	6.73%	0.00%
5	105535.89	105535.89	108192.80	2.46%	2.46%	0.00%
6	94011.72	94011.72	96680.70	2.76%	2.76%	0.00%
7	147965.46	147965.46	158635.20	6.73%	6.73%	0.00%
8	107165.46	107165.46	108993.60	1.68%	1.68%	0.00%
9	94011.72	94011.72	96680.70	2.76%	2.76%	0.00%
10	147965.46	147965.46	158635.20	6.73%	6.73%	0.00%
11	107165.46	107165.46	108993.60	1.68%	1.68%	0.00%
12	94011.72	94011.72	96680.70	2.76%	2.76%	0.00%
13	158465.46	158465.46	167550.50	5.42%	5.42%	0.00%
14	117665.46	117665.46	119415.20	1.47%	1.47%	0.00%
15	109761.72	109761.72	111975.40	1.98%	1.98%	0.00%
16	158465.46	158465.46	167550.50	5.42%	5.42%	0.00%
17	117665.46	117665.46	119415.20	1.47%	1.47%	0.00%
18	109761.72	109761.72	111975.40	1.98%	1.98%	0.00%
19	158465.46	158465.46	167550.50	5.42%	5.42%	0.00%
20	117665.46	117665.46	119415.20	1.47%	1.47%	0.00%
21	109761.72	109761.72	111975.40	1.98%	1.98%	0.00%
22	158465.46	158465.46	167550.50	5.42%	5.42%	0.00%
23	117665.46	117665.46	119415.20	1.47%	1.47%	0.00%
24	109761.72	109761.72	111975.40	1.98%	1.98%	0.00%
Average Gap				3.40%	3.40%	0.00%

Table 5.12: Lower bounds for scenario 4 after valid inequality addition , Partial LP .

Parameter settings	Lower Bounds	Lower Bounds after Covers	Optimal	Gap From Optimal No Cuts	Gap From Optimal with Cuts	Improvement of Gaps
1	147965.46	147965.46	160457.30	7.79%	7.79%	0.00%
2	104603.93	104603.93	110814.80	5.60%	5.60%	0.00%
3	94011.72	94011.72	100685.20	6.63%	6.63%	0.00%
4	147965.46	147965.46	161405.50	8.33%	8.33%	0.00%
5	105535.89	105535.89	111884.40	5.67%	5.67%	0.00%
6	94011.72	94011.72	100685.20	6.63%	6.63%	0.00%
7	147965.46	147965.46	168593.90	12.24%	12.24%	0.00%
8	107165.46	107165.46	119128.70	10.04%	10.04%	0.00%
9	94011.72	94011.72	100685.20	6.63%	6.63%	0.00%
10	147965.46	147965.46	169565.50	12.74%	12.74%	0.00%
11	107165.46	107165.46	120088.40	10.76%	10.76%	0.00%
12	94011.72	94011.72	100685.20	6.63%	6.63%	0.00%
13	158465.46	158465.46	171362.50	7.53%	7.53%	0.00%
14	117665.46	117665.46	124261.10	5.31%	5.31%	0.00%
15	109761.72	109761.72	114577.30	4.20%	4.20%	0.00%
16	158465.46	158465.46	172345.30	8.05%	8.05%	0.00%
17	117665.46	117665.46	125353.10	6.13%	6.13%	0.00%
18	109761.72	109761.72	114577.30	4.20%	4.20%	0.00%
19	158465.46	158465.46	179522.50	11.73%	11.73%	0.00%
20	117665.46	117665.46	132421.10	11.14%	11.14%	0.00%
21	109761.72	109761.72	114577.30	4.20%	4.20%	0.00%
22	15465.46	15465.46	180505.30	91.43%	91.43%	0.00%
23	117665.46	117665.46	133513.10	11.87%	11.87%	0.00%
24	109761.72	109761.72	114577.30	4.20%	4.20%	0.00%
Average Gap				11.24%	11.24%	0.00%

However cover inequalities have improved lower bounds from full LP relaxation slightly, there is no change observed in partial LP relaxation results. To improve the efficiency of the cover inequalities the number of covers added to each problem can be increased. We think generating higher number of cover sets will affect the lower bounds from both full and partial LP relaxations. Summary of average gaps and improvements from full LP and partial LP bounds through cover inequality addition is given below in Table 5.13.

Table 5.13: Summary of LP relaxation results.

	Scenarios			
	1	**2**	**3**	4
	Gap from optimal	**Gap from optimal**	**Gap from optimal**	**Gap from optimal**
Full LP	14.12%	15.62%	20.45%	24.18%
Full LP with cover inequalities	14.11%	15.61%	20.17%	23.96%
Partial LP	1.91%	3.67%	3.40%	11.24%
Partial LP with cover inequalities	1.91%	3.67%	3.40%	11.24%
Full LP gap improvement	0.03%	0.03%	0.73%	0.54%
Partial LP gap improvement	0.00%	0.00%	0.00%	0.00%

6. SUGGESTED HEURISTIC METHOD

Our problem is NP-Hard, which needs an excessive amount of solution time for large scale problems. The heuristic method suggested here is a "fix and optimize heuristic" that has two main phases. In the first phase, a feasible and quality solution is found, and in the second phase an improvement process is executed on this solution. The main logic here is that the problem is divided into smaller parts, and it is solved in the form of partial integer programming problems.

First phase of the heuristic method for the case of our problem consists of seven steps as given below. At first step, the problem is divided to P smaller sub problems based on customer indices. Next, the first sub problem's variables are defined as binaries and the other (P -1) sub problems are left as linearly relaxed problems, which means that all the binary variables are defined in the [0,1] interval and the integer variables are defined as $R_{\geq 0}$. In the case of our model, the only variable which has the index i which demonstrates customer numbers, is X_{ivf} and thus we define these variables as binaries in each sub problem iteratively and fix the binary solutions for them. In addition to X_{ivf}, the variable related to the ownership cost, i.e. vehicle usage indicator variable V_v are defined as a binary variable in all of the steps during the first phase. After defining first sub problem's X_{ivf} and V_v as binaries, the main problem is solved. In the second step, the values for binary variables from the first step solution is fixed and X_{ivf} of the next sub problem is defined as binaries. The rest (P- 2) sub problems are solved as relaxed problems. This process continues until all the X_{ivf} variables for the P sub problems become binaries. At this step, the X_{ivf} values are fixed and all the other variables are re-defined as binaries and the main problem is solved another time. At the end of these five steps a feasible integer solution will be in hand. Steps of the heuristic's first phase are given below:

6.1 First Phase

Step 1: Determine the number of customers based on which the problem is divided into sub problems. Divide the decision variables to P parts based on the same criterion. Let A_i be the set of binary variables in ith sub problem.

Step 2: Set the iterator to zero (i = 0).

Step 3: If i $>$ 0 , set the binaries in A_j to the values of variables in iteration(i-1) for J = 1…i-1

Step 4: Relax the binary variables in A_j for J = i+1 …P .

Step 5: Define the variables in S_i as binaries.

Step 6: Solve the model and set i to i+1. If i$<$ P go to step 3. If i =Z go to step 7.

Step 7: Re-define all the binaries except X_{ivf} and resolve the problem using the X_{ivf} values from iteration i = Z-1.

At the end of first phase, a quality feasible solution is found. In the second phase the heuristic tries to improve the solution from first phase. In this procedure no variables are relaxed and all of the variables are defined either as binaries or are fixed to the values from previous step. At the beginning of phase two the counter i is set to zero, i.e. i = 0. Then the values of X_{ivf} 's from first phase are cleared and are re-defined as binaries for ith sub problem. Next X_{ivf} variables for the rest of (P-1) sub problems are fixed to the values from the first phase. After these settings the main problem is solved. In the next step, setting X_{ivf} values to the solution from the last step, we define all the other variables of ith problem as binaries or integers and we re-solve the main problem. Let's call the last step in which we define all the variables as binaries "binarizer". In the second step, we define the X_{ivf} variables of the next sub problem as binaries and get the other X_{ivf} values from previously fixed solutions. After re-defining the X_{ivf}s and solving the sub problem, at each iteration we use the binarizer step to force all the variables to be binaries/integers. Except for step one, after each binarizer step we compare the new objective value to the last objective

value we found from previous solutions. If any improvements occur, we fix the X_{ivf} values, go back to the first sub problem and re-define X_{ivf} and solve the problem again. If no improvements occur, no backward movements are done. Steps of second phase are explained below.

6.2 Second Phase

Step 1: Set the counter i to 0

Step2: Clear the values of all variables in A_i and re-define them as binaries

Step 3: For j =1,2,..,P and j≠i set the variable values to previously found solutions.

Step 4: Solve the model and fix X_{ivf} values, re-solve the model defining all the variables as binaries and getting X_{ivf} values from the last solution, set i =i+1

Step 5: Check the objective value, if improvements occurred and i< P, update the objective value and go to step 2. If no improvement occurred and i< P go to step 2. If i = P stop.

7. PERFORMANCE ANALYSIS OF THE SUGGESTED HEURISTIC

The heuristic approach which is suggested above was applied to the problem using Eclipse Java, and mathematical models were solved using Concert Technology with Java. We solved the problem in presence and absence of valid inequalities for assessing the effect of the cuts we generated previously. For some parameter settings the valid inequalities could decrease the gaps between optimal solution and solution form heuristic application. For the cases which we could not get optimal solutions with CPLEX, we compared the heuristic solutions with the best LP bound from CPLEX. Solution times decreased significantly and all of the parameter settings were solved under 20 seconds. In general, the gaps between optimal solution and heuristic solutions were tight but in some cases we had large gaps. In order to decrease these gaps we changed the order of sub problem selection which led into reductions for some problems. All the results are presented in Table 7.1 to Table 7.8.

Table 7.1: Heuristic results for Scenario 1 without cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**
1	115021.11	115021.11	113304.72	1.51%
2	92472.44	92472.44	91365.28	1.21%
3	82783.72	82783.72	81868.77	1.12%
4	116331.51	116331.51	114578.72	1.53%
5	93220.81	93220.81	91503.29	1.88%
6	82783.72	82783.72	81868.77	1.12%
7	123181.11	123181.11	121464.72	1.41%
8	93886.58	93886.58	91503.29	2.60%
9	82776.07	82776.07	81868.77	1.11%
10	123214.90	123214.90	122738.72	0.39%
11	92472.44	92472.44	91503.29	1.06%
12	82783.72	82783.72	81868.77	1.12%
13	121999.82	121999.82	120527.44	1.22%
14	102674.87	102674.87	101778.94	0.88%
15	96934.10	96934.10	95981.98	0.99%
16	122243.44	122243.44	121801.44	0.36%
17	102232.87	102232.87	101778.94	0.45%
18	97087.10	97087.10	95981.98	1.15%
19	130159.82	130159.82	128687.44	1.14%
20	104049.79	104049.79	101778.94	2.23%
21	96935.10	96935.10	95981.98	0.99%
22	130403.44	130403.44	129961.44	0.34%
23	102689.62	102689.62	101778.94	0.89%
24	97087.10	97087.10	95981.98	1.15%
Average Gap				1.16%

Table 7.2: Heuristic results for scenario2 without cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**	**Gap from best LP**
1	114788.66	114788.66	114788.66	0.00%	
2	96909.51	96909.51	92188.28	5.12%	
3	83268.44	83268.44	83268.44		2.86%
4	116099.06	116099.06	116099.06	0.00%	
5	96909.51	96909.51	92846.28		0.84%
6	83268.44	83268.44	83268.44		2.92%
7	122948.66	122948.66	122948.66	0.00%	
8	96909.51	96909.51	95850.94		2.87%
9	83268.44	83268.44	83268.44		1.97%
10	124259.06	124259.06	124259.06	0.00%	
11	96909.51	96909.51	95850.94		3.85%
12	83268.44	83268.44	83268.44		2.01%
13	121976.93	121976.93	121976.93	0.00%	
14	106449.97	106449.97	105218.27		2.26%
15	98211.87	98211.87	96341.47		0.82%
16	123287.33	123287.33	123287.33	0.00%	
17	106449.97	106449.97	105489.51		1.96%
18	98211.87	98211.87	96341.47		1.00%
19	130136.93	130136.93	130136.93	0.00%	
20	106449.97	106449.97	105489.51		2.53%
21	98211.87	98211.87	96341.47	1.94%	
22	131447.33	131447.33	131447.33	0.00%	
23	106449.97	106449.97	105489.51		2.37%
24	98211.87	98211.87	96341.47	1.94%	
Average Gap				0.82%	2.17%

Table 7.3: Heuristic results for scenario3 without cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**
1	198942.24	198942.24	158635.20	25.41%
2	109841.04	109841.04	107246.40	2.42%
3	98257.17	98257.17	96680.70	1.63%
4	194883.34	194883.34	158635.20	22.85%
5	109306.46	109306.46	108192.80	1.03%
6	98176.22	98176.22	96680.70	1.55%
7	207102.24	207102.24	158635.20	30.55%
8	146473.24	146473.24	108993.60	34.39%
9	97573.66	97573.66	96680.70	0.92%
10	207228.24	207228.24	158635.20	30.63%
11	146762.24	146762.24	108993.60	34.65%
12	98494.90	98494.90	96680.70	1.88%
13	211345.06	211345.06	167550.50	26.14%
14	139040.89	139040.89	119415.20	16.43%
15	113042.43	113042.43	111975.40	0.95%
16	212447.46	212447.46	167550.50	26.80%
17	120165.12	120165.12	119415.20	0.63%
18	113484.43	113484.43	111975.40	1.35%
19	220425.47	220425.47	167550.50	31.56%
20	159145.49	159145.49	119415.20	33.27%
21	113042.43	113042.43	111975.40	0.95%
22	220287.22	220287.22	167550.50	31.48%
23	159145.46	159145.46	119415.20	33.27%
24	113627.27	113627.27	111975.40	1.48%
Average Gap				16.34%

Table 7.4: Heuristic results for scenario 4 without cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**	**Gap from best LP**
1	193845.87	193845.87	160457.30		22.19%
2	110969.37	110969.37	110814.80		2.62%
3	101700.26	101700.26	100685.20		3.90%
4	194223.87	194223.87	161405.50	20.33%	
5	112027.77	112027.77	111884.40		2.87%
6	101700.26	101700.26	100685.20		3.96%
7	202005.87	202005.87	168593.90		20.03%
8	119129.37	119129.37	119128.70		2.84%
9	101700.26	101700.26	100685.20		3.00%
10	202383.87	202383.87	169565.50	19.35%	
11	120187.77	120187.77	120088.40		3.87%
12	101700.26	101700.26	100685.20		3.04%
13	208987.13	208987.13	171362.50	21.96%	
14	124261.15	124261.15	124261.10		1.12%
15	115009.15	115009.15	114577.30		1.20%
16	209387.53	209387.53	172345.30		22.25%
17	125353.15	125353.15	125353.10		1.16%
18	115009.15	115009.15	114577.30		1.38%
19	180256.40	180256.40	179522.50		1.02%
20	132421.15	132421.15	132421.10		0.93%
21	115009.15	115009.15	114577.30	0.38%	
22	217547.53	217547.53	180505.30		20.59%
23	133513.15	133513.15	133513.10		0.80%
24	115009.15	115009.15	114577.30	0.38%	
Average Gap				12.48%	6.25%

Table 7.5: Heuristic results for scenario1 with cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**	**improvement with cuts**
1	113746.72	113746.72	113304.70	0.39%	1.12%
2	93220.81	93220.81	91365.28	2.03%	-0.80%
3	82783.72	82783.72	81868.77	1.12%	0.00%
4	115020.72	115020.72	114578.70	0.39%	1.14%
5	92472.44	92472.44	91503.29	1.06%	0.81%
6	82783.72	82783.72	81868.77	1.12%	0.00%
7	121906.72	121906.72	121464.70	0.36%	1.05%
8	92506.51	92506.51	91503.29	1.10%	1.49%
9	82783.72	82783.72	81868.77	1.12%	-0.01%
10	123180.72	123180.72	122738.70	0.36%	0.03%
11	93302.50	93302.50	91503.29	1.97%	-0.89%
12	82783.72	82783.72	81868.77	1.12%	0.00%
13	120969.44	120969.44	120527.40	0.37%	0.85%
14	102743.02	102743.02	101778.90	0.95%	-0.07%
15	96781.10	96781.10	95981.98	0.83%	0.16%
16	122243.44	122243.44	121801.40	0.36%	0.00%
17	102743.02	102743.02	101778.90	0.95%	-0.50%
18	97087.10	97087.10	95981.98	1.15%	0.00%
19	129129.44	129129.44	128687.40	0.34%	0.80%
20	102743.02	102743.02	101778.90	0.95%	1.27%
21	96781.10	96781.10	95981.98	0.83%	0.16%
22	130403.44	130403.44	129961.40	0.34%	0.00%
23	102743.02	102743.02	101778.90	0.95%	-0.05%
24	97087.10	97087.10	95981.98	1.15%	0.00%
Average Gaps				0.89%	0.27%

Table 7.6: Heuristic results for scenario2 with cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**	**Gap from best LP**	**improvement with cuts**
1	114788.66	114788.66	114788.70	0.00%		0.00%
2	96909.51	96909.51	92188.28	5.12%		0.00%
3	83268.44	83268.44	83268.44		2.86%	0.00%
4	116099.06	116099.06	116099.10	0.00%		0.00%
5	96909.51	96909.51	92846.28		0.84%	0.00%
6	83268.44	83268.44	83268.44		2.92%	0.00%
7	122948.66	122948.66	122948.70	0.00%		0.00%
8	96909.51	96909.51	95850.94		2.87%	0.00%
9	83268.44	83268.44	83268.44		1.97%	0.00%
10	124259.06	124259.06	124259.10	0.00%		0.00%
11	96909.51	96909.51	95850.94		3.85%	0.00%
12	83268.44	83268.44	83268.44		2.01%	0.00%
13	121976.93	121976.93	121976.90	0.00%		0.00%
14	106449.97	106449.97	105218.30		2.26%	0.00%
15	98211.87	98211.87	96341.47		0.82%	0.00%
16	123287.33	123287.33	123287.30	0.00%		0.00%
17	106449.97	106449.97	105489.50		1.96%	0.00%
18	98058.87	98058.87	96341.47		1.00%	0.16%
19	130136.93	130136.93	130136.90	0.00%		0.00%
20	106449.97	106449.97	105489.50		2.53%	0.00%
21	98211.87	98211.87	96341.47	1.94%		0.00%
22	131447.33	131447.33	131447.30	0.00%		0.00%
23	106449.97	106449.97	105489.50		2.37%	0.00%
24	98058.87	98058.87	96341.47	1.78%		0.16%
Average Gaps				0.80%	2.17%	0.01%

Table 7.7: Heuristic results for scenario3 with cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**	**improvement with cuts**
1	198942.24	198942.24	158635.24	25.41%	0.00%
2	109399.04	109399.04	107246.37	2.01%	0.40%
3	97573.66	97573.66	96680.70	0.92%	0.70%
4	199068.24	199068.24	158635.24	25.49%	-2.15%
5	109728.31	109728.31	108192.77	1.42%	-0.39%
6	98618.22	98618.22	96680.70	2.00%	-0.45%
7	206218.24	206218.24	158635.24	30.00%	0.43%
8	146762.24	146762.24	108993.58	34.65%	-0.20%
9	97573.66	97573.66	96680.70	0.92%	0.00%
10	207228.24	207228.24	158635.24	30.63%	0.00%
11	146762.24	146762.24	108993.58	34.65%	0.00%
12	97573.66	97573.66	96680.70	0.92%	0.94%
13	211345.06	211345.06	167550.49	26.14%	0.00%
14	120165.12	120165.12	119415.17	0.63%	13.58%
15	112877.31	112877.31	111975.40	0.81%	0.15%
16	212447.46	212447.46	167550.49	26.80%	0.00%
17	120165.12	120165.12	119415.17	0.63%	0.00%
18	113513.57	113513.57	111975.40	1.37%	-0.03%
19	220425.47	220425.47	167550.49	31.56%	0.00%
20	159145.49	159145.49	119415.17	33.27%	0.00%
21	112877.31	112877.31	111975.40	0.81%	0.15%
22	220607.46	220607.46	167550.49	31.67%	-0.15%
23	120165.12	120165.12	119415.17	0.63%	24.49%
24	113008.66	113008.66	111975.40	0.92%	0.54%
Average Gaps				14.34%	1.58%

Table 7.8: Heuristic results for scenario4 with cover inequalities.

Parameter settings	**First Step**	**Second Step**	**Optimal**	**Gap from Optimal**	**Gap from best LP**	**improvement with cuts**
1	193845.87	193845.87	160457.25		22.19 %	0.00%
2	110969.37	110969.37	110814.77		2.62%	0.00%
3	101700.26	101700.26	100685.25		3.90%	0.00%
4	194223.87	194223.87	161405.53	20.33%		0.00%
5	112027.77	112027.77	111884.37		2.87%	0.00%
6	101700.26	101700.26	100685.25		3.96%	0.00%
7	169383.40	169383.40	168593.93		0.65%	16.15%
8	119129.37	119129.37	119128.74		2.84%	0.00%
9	101700.26	101700.26	100685.25		3.00%	0.00%
10	202383.87	202383.87	169565.53	19.35%		0.00%
11	120187.77	120187.77	120088.45		3.87%	0.00%
12	101700.26	101700.26	100685.25		3.04%	0.00%
13	208987.13	208987.13	171362.55	21.96%		0.00%
14	124261.15	124261.15	124261.15		1.12%	0.00%
15	115009.15	115009.15	114577.27		1.20%	0.00%
16	209387.53	209387.53	172345.35		22.25 %	0.00%
17	125353.15	125353.15	125353.15		1.16%	0.00%
18	115009.15	115009.15	114577.27		1.38%	0.00%
19	217147.13	217147.13	179522.55		21.70 %	-20.47%
20	132421.15	132421.15	132421.15		0.93%	0.00%
21	115009.15	115009.15	114577.27	0.38%		0.00%
22	181020.80	181020.80	180505.35		0.35%	16.79%
23	133513.15	133513.15	133513.15		0.80%	0.00%
24	115009.15	115009.15	114577.27	0.38%		0.00%
Average Gap				12.48%	5.25%	0.52%

As it is shown in the tables, in general the heuristic method we utilized for solving our problem yields good and near optimal solutions for large number of parameter settings. For example, in the case without the addition of cover inequalities in scenario 2 with parameter settings 1, 4, 7, 10, 16, 19 and 22 we could find the optimal solution of the problem. After addition of cover inequalities of four different types, we can see the difference between solutions in absence of valid inequalities and the ones in presence of them. To illustrate more, in scenario 3 with parameter setting 14, we have an improvement of 13.58% from the solution without cover inequalities and in the same scenario with parameter setting of 23 we have 24.49% of improvement in comparison to the results gained in absence of valid inequalities.

One other scenario in which we met an acceptable improvement through cover inequalities addition is scenario 4 in which for parameter settings 7 and 22 we have 16.15% and 16.79% of improvement respectively.

Along with these improvements, in a few cases we have high gaps such as the 33.27% gap in scenario 3 with parameter setting 20. Summary of the results of heuristic approach application is given below in Table 7.9.

Table 7.9: Result of heuristic method.

	Scenarios			
	1	**2**	**3**	4
	Gap from optimal	**Gap from optimal**	**Gap from optimal**	**Gap from optimal**
Without cover inequalities	1.16%	1.55%	16.34%	7.55%
Avr. max and min gap	(2.60%, 0.34%)	(5.12%,0.0%)	(34.65%,0.63%)	(22.25%,0.63%)
With cover inequalities	0.89%	1.55%	14.34%	6.67%
Avr., max and min gap	(2.03%,0.34%)	(5.12%,0.0%)	(34.65%,0.63%)	(22.25%,0.63%)

After heuristic application we could get results which have gaps under 1% for all scenarios. The number of problems which had gaps under 1% with cover inequality addition and without cover inequality addition is summarized in Table 7.10.

Table 7.10: Number of solutions with gaps under 1%.

	Number of gaps under 1% without covers	**Number of gaps under 1% with covers**
Scenario1	8	15
Scenario2	11	11
Scenario3	4	9
Scenario4	4	6

To handle high gap problem we changed the order of fixation in alternative ways. Instead of starting solution from first customer we start from different customer groups. To exemplify, the solution process started from customer group 4 and respectively went through 2, 3 and 1. We also tested following there combinations as the additional combinations. The order of fixation in each random combination is given in Table 7.11. The results gained through these random fixation orders are presented in Table 7.12 throug Table 7.15.

Table 7.11: Table of random fixations.

Order of fixation	**Number of combinations**
4-2-3-1	1
3-4-2-1	2
2-3-4-1	3

Table 7.12: Results of Scenario 1 with random combinations.

Parameter settings	**Gap from Optimal/LB, Fixation order1**	**Gap from Optimal/LB, Fixation order2**	**Gap from Optimal/LB, Fixation order3**
1	0.39%	0.47%	1.51%
2	2.12%	1.21%	2.12%
3	1.47%	1.17%	1.20%
4	0.39%	1.52%	0.39%
5	2.45%	1.79%	39.84%
6	1.17%	1.18%	13.38%
7	1.40%	0.37%	0.37%
8	1.97%	1.12%	1.97%
9	1.47%	1.17%	2.63%
10	0.36%	1.42%	0.36%
11	2.45%	2.37%	1.20%
12	1.17%	1.18%	2.66%
13	0.37%	0.50%	0.50%
14	1.50%	1.78%	18.67%
15	1.15%	0.99%	0.99%
16	0.37%	0.37%	0.37%
17	1.50%	0.98%	1.93%
18	0.99%	1.15%	1.15%
19	1.13%	0.35%	0.35%
20	1.50%	1.78%	1.50%
21	1.15%	1.15%	0.99%
22	0.35%	0.35%	0.35%
23	1.50%	0.98%	1.93%
24	0.99%	1.15%	1.15%
Average Gaps	1.22%	1.10%	4.06%

Table 7.13 : Results of Scenario 2 with random combinations.

Parameter settings	**Gap from Optimal/LB, Fixation order1**	**Gap from Optimal/LB, Fixation order2**	**Gap from Optimal/LB, Fixation order3**
1	0.00%	0.00%	0.00%
2	5.12%	4.02%	5.18%
3	2.86%	2.86%	4.87%
4	0.00%	0.00%	0.00%
5	0.84%	0.84%	5.32%
6	2.92%	2.92%	4.94%
7	0.00%	0.00%	0.00%
8	2.87%	2.87%	4.07%
9	1.97%	1.97%	3.97%
10	0.00%	0.00%	0.00%
11	3.85%	3.85%	5.06%
12	2.01%	2.01%	4.01%
13	0.00%	0.00%	0.00%
14	2.26%	2.26%	3.34%
15	0.82%	0.82%	1.07%
16	0.00%	0.00%	0.00%
17	1.96%	1.96%	2.77%
18	1.00%	1.00%	1.25%
19	0.00%	0.00%	0.00%
20	2.53%	2.53%	3.35%
21	0.24%	1.78%	0.24%
22	0.00%	0.00%	0.00%
23	2.37%	2.37%	3.18%
24	0.24%	1.78%	0.24%
Average Gaps	1.41%	1.49%	2.20%

Table 7.14: Results of Scenario 3 with random combinations.

Parameter settings	**Gap from Optimal/LB, Fixation order1**	**Gap from Optimal/LB, Fixation order2**	**Gap from Optimal/LB, Fixation order3**
1	21.40%	26.63%	21.16%
2	1.45%	2.48%	1.01%
3	1.00%	1.60%	1.68%
4	22.85%	26.12%	22.80%
5	2.11%	2.11%	2.11%
6	1.03%	0.91%	1.00%
7	22.85%	21.16%	24.74%
8	2.03%	1.01%	1.01%
9	1.68%	1.01%	1.68%
10	22.85%	22.80%	21.12%
11	19.63%	1.01%	1.01%
12	1.03%	0.91%	39.63%
13	21.51%	20.07%	21.51%
14	1.22%	33.95%	1.22%
15	1.18%	0.79%	0.94%
16	21.60%	20.34%	20.01%
17	0.60%	1.22%	1.22%
18	0.94%	1.38%	0.99%
19	21.51%	21.51%	21.51%
20	1.22%	33.95%	1.22%
21	1.18%	1.38%	0.94%
22	21.60%	21.60%	21.60%
23	0.60%	1.22%	1.22%
24	0.94%	1.38%	0.99%
Average Gaps	8.92%	11.11%	9.68%

Table 7.15 : Results of Scenario 4 with random combinations.

Parameter settings	**Gap from Optimal/LB, Fixation order1**	**Gap from Optimal/LB, Fixation order2**	**Gap from Optimal/LB, Fixation order3**
1	1.54%	22.12%	22.04%
2	2.62%	2.62%	3.46%
3	3.90%	3.38%	3.76%
4	20.31%	20.27%	20.19%
5	2.87%	2.88%	3.74%
6	3.96%	3.44%	3.82%
7	20.01%	19.97%	19.90%
8	2.84%	2.84%	3.63%
9	3.00%	2.48%	2.87%
10	19.33%	19.29%	0.26%
11	3.87%	3.87%	4.68%
12	3.04%	2.52%	2.91%
13	21.91%	21.79%	21.71%
14	1.12%	1.28%	1.54%
15	1.20%	0.82%	1.08%
16	22.20%	22.08%	0.88%
17	1.16%	1.32%	1.61%
18	1.38%	1.00%	1.26%
19	21.65%	21.54%	21.46%
20	0.93%	1.08%	1.33%
21	0.38%	0.00%	0.25%
22	20.55%	0.31%	20.36%
23	0.80%	0.95%	1.22%
24	0.38%	0.00%	0.25%
Average Gaps	7.54%	7.41%	6.84%

As it is presented in the tables, the order of fixation operation affects the amount of gaps from optimal solution. Summary of best gaps we got from multiple fixation orders is presented below:

Table 7.16: Summary of best gaps through heuristic application.

	Scenarios			
	1	**2**	**3**	**4**
	Gap from optimal/LB	**Gap from optimal/LB**	**Gap from optimal/LB**	**Gap from optimal/LB**
Best average gap	0.81%	1.37%	7.67%	4.04%
Min gap	0.34%	0.00%	0.60%	0.00%
Max gap	1.21%	4.02%	22.80%	21.71%
Number of solutions with gaps under 1%	16	12	11	9

8. CONCLUSION

In this research, we studied a problem of integrated replenishment planning and fleet sizing in a vendor managed inventory system. The decisions to be made are the assignment of vehicles and frequencies to the customers, which have deterministic demand, along with the fleet composition. As mentioned in the section of problem definition, demand divisibility is not considered in this problem and the demand of each customer can be shipped using only single vehicle and a single frequency. We have a given set of discrete frequencies based on weeks and a single frequency for daily replenishments. The model used to represent the problem is a full integer formulation. The introduced mathematical model of the problem itself is a new contribution to the literature

The problem solutions are examined and analyzed under four different scenarios including no clusters/ four clusters with normal and 50% higher demand. Analyzing the results of decision variables under different cases we showed some logical relations between different factors and their effects on the solution. As seen in the computational results section, grouping customers into different categories directly affects all costs. In the case with no clusters and normal demand, the possibility of demand consolidation is high resulting in less frequent replenishments and higher inventory holding costs versus lower transportation costs due to less number of replenishments. Grouping into four categories, brings the limitation for some customers that are not allowed to be in the same route. This situation results in higher frequencies, lower lot sizes, higher transportation costs, and also lower inventory costs. When clusters are formed, frequencies are reduced in order to create more consolidation possibility. The reduction in replenishment frequencies will result in lower dead heading and replenishment costs.

An additional chance of consolidation is provided by shifting from small vehicles to large vehicles with the addition of the clusters. If the replenishment frequencies did

not change with the cluster addition, there would be a need for more vehicles, which would cause the ownership and routing costs to grow very high. To avoid this cost explosion, frequencies are lowered and lots are adjusted to be shipped in larger vehicles. In scenarios with larger vehicles (capacities of 21-27), this consolidation of clustered case can be handled using a less number of large vehicles. For the same case, in some scenarios only small vehicles are used. This change is occurred because the capacity of small vehicles is large enough for handling the consolidated demands under the mentioned scenarios.

Generally, cluster addition causes frequency increases and thus less frequent replenishments will be performed. As a result, the lots carried at each replenishment will be larger in comparison to the cases in which customers are not divided into clusters. This is the reason to route number reduction in clustered scenarios.. This reduced route number will reduce dead heading costs. Considering lower replenishment frequencies (less repetitions of fixed per kilometer cost) and lower dead heading costs we expect the routing costs reduction. Routing costs are composed of dead heading costs and fixed per kilometer costs, which are higher for the larger vehicles. This fact makes it possible to explain the reason for increase in routing costs when customers are grouped. A shift from smaller vehicles to larger vehicles causes this increase.

In thist, the suggested problem is NP-hard and we introduced a fix and optimize type solution procedure. Generally the heuristic results showed an acceptable gap from optimal/ best LP bound, except for a very few parameter settings. The problem of high gaps from optimality was reduced through changing the order of fixation in steps of the heuristic approach. As a result of this adjustment, we could successfully decrease further the gaps of specific parameter settings. Checking table 7.28, we can see that, overall, the average optimality gap we obtained in 96 problems is 3.47% and half of the problems are solved with less than 1% optimality gap.

In summary, we contributed to the literature in three aspects; 1. We suggested a mathematical model for this important problem. 2. We carried out some numerical analysis and drawn some insights. 3. We developed an effective optimization based heuristic approach as a solution procedure.

REFRENCES

Adulyasak, Y., Cordeau, J. Jans, R., (2015), "The production routing problem: A review of formulations and solution algorithms, ", Computers & Operations Research, Vol. 55, pp. 141-152.

Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a Heterogeneous Fleet of Vehicles. In B. Golden, S. Raghavan & E. Wasil (Eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (Vol. 43, pp. 3-27): Springer US

Bertazzi, L., & Grazia Speranza, M. (1999). Inventory control on sequences of links with given transportation frequencies. *International Journal of Production Economics, 59*(1–3), 261-270. doi: http://dx.doi.org/10.1016/S0925-5273(98)00235-7

Bertazzi, L., Paletta, G., & Speranza, M. G. (2005). Minimizing the Total Cost in an Integrated Vendor—Managed Inventory System. *Journal of Heuristics, 11*(5-6), 393-419. doi: 10.1007/s10732-005-0616-6.

Bertazzi, L., & Speranza, M. G. (2002). Continuous and Discrete Shipping Strategies for the Single Link Problem. *Transportation Science, 36*(3), 314-325. doi: doi:10.1287/trsc.36.3.314.7828.

Bertazzi, L., Speranza, M. G., & Ukovich, W. (1997). Minimization of logistic costs with given frequencies. *Transportation Research Part B: Methodological, 31*(4), 327-340. doi: http://dx.doi.org/10.1016/S0191-2615(96)00029-X

Boudia, M., Louly, M. A. O., Prins, C.,(2007), "A reactive GRASP and path relinking for a combined production-distribution problem", Computers and operations research, Vol. 34, No. 11, pp. 3402-3419.

Chandra, Pankaj, and Marshall L. Fisher. (1994): "Coordination of production and distribution planning." *European Journal of Operational Research* 72.3 503-517.

Coelho, C., Cordeou, J., Laporte, G.,(2013), "Thirty Years of Inventory Routing", Vol. 48, No. 1, pp. 1-19.

Coelho, C., Laporte, G., (2013), "The exact solution of several classes of inventory-routing problems", Vol. 40,No. 2, pp. 558-565.

Coffman Jr, Edward G., Michael R. Garey, and David S. Johnson. (1996) "Approximation algorithms for bin packing: a survey." *Approximation algorithms for NP-hard problems*. PWS Publishing Co.

Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research, 122*(2), 272-288. doi: http://dx.doi.org/10.1016/S0377-2217(99)00233-7

Desrochers, M., & Verhoog, T. W. (1991). A new heuristic for the fleet size and mix vehicle routing problem. *Computers & Operations Research, 18*(3), 263-274. doi: http://dx.doi.org/10.1016/0305-0548(91)90028-P

Dorneles, Á. P., de Araújo, O. C. B., & Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers & Operations Research, 52, Part A*, 29-38. doi: http://dx.doi.org/10.1016/j.cor.2014.06.023

Federgruen, Awi, Joern Meissner, and Michal Tzur. (2007): "Progressive interval heuristics for multi-item capacitated lot-sizing problems." *Operations Research*55.3 490-502.

Fleszar, Krzysztof, and Khalil S. Hindi. (2002) "New heuristics for one-dimensional bin-packing." *Computers & operations research* 29.7: 821-839.

E. G. Coffman, J., Garey, M. R., & Johnson, D. S. (1997). Approximation algorithms for bin packing: a survey. In S. H. Dorit (Ed.), *Approximation algorithms for NP-hard problems* (pp. 46-93): PWS Publishing Co.

Ertogral, K., & Gonzalez, S. (2015). Modelling and Analysis of a Strategic Fleet Sizing Problem for a Furniture Distributor: TOBB University of Economics And Technology.

Federgruen, A., Meissner, J., & Tzur, M. (2007). Progressive Interval Heuristics for Multi-Item Capacitated Lot-Sizing Problems. *Operations Research, 55*(3), 490-502. doi: doi:10.1287/opre.1070.0392.

Gintner, Vitali, Natalia Kliewer, and Leena Suhl. (2005)."Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice." *OR Spectrum* 27.4: 507-523.

Hakan Keskin, "Supply chain management" Nobel Yayınları .

Helber, Stefan, and Florian Sahling. (2010) "A fix-and-optimize approach for the multi-level capacitated lot sizing problem." *International Journal of Production Economics* 123.2: 247-256.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., & Løkketangen, A. (2010). Industrial aspects and literature survey: Fleet composition and routing.

Computers & Operations Research, *37*(12), 2041-2061. doi: http://dx.doi.org/10.1016/j.cor.2010.03.015

Hohmann, S., & Zelewski, S. (2012). Effects of vendor-managed inventory on the bullwhip effect. *Management Innovations for Intelligent Supply Chains*, 167.

Huang, M., Smilowitz, K. R., & Balcik, B. (2013). A continuous approximation approach for assessment routing in disaster relief. *Transportation Research Part B: Methodological*, *50*(0), 20-41. doi: http://dx.doi.org/10.1016/j.trb.2013.01.005

Hvattum, Lars Magnus, and Arne Løkketangen. (2009) "Using scenario trees and progressive hedging for stochastic inventory routing problems." *Journal of Heuristics* 15.6: 527-557.

Jabali, O., Gendreau, M., & Laporte, G. (2012). A continuous approximation model for the fleet composition problem. *Transportation Research Part B: Methodological*, *46*(10), 1591-1606. doi: http://dx.doi.org/10.1016/j.trb.2012.06.004

Jaillet, Patrick, et al. (2002) "Delivery cost approximations for inventory routing problems in a rolling horizon framework." *Transportation Science* 36.3: 292-300.

Karmarkar, Narendra, and Richard M. Karp. (1982)"An efficient approximation scheme for the one-dimensional bin-packing problem." *Foundations of Computer Science, SFCS'08. 23rd Annual Symposium on*. IEEE.

Kröger, Berthold. (1995) "Guillotineable bin packing: A genetic approach." *European Journal of Operational Research* 84.3: 645-661.

Leandro C. Coelho, J.-F. C., Gilbert Laporte. (2014). Thirty Years of Inventory Routing. *Transportation Science*, *48*(1), 1-19. doi: doi:10.1287/trsc.2013.0472.

Lodi, Andrea, Silvano Martello, and Daniele Vigo. (1999) "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems." *INFORMS Journal on Computing* 11.4: 345-357.

Luca Bertazzi, M. G. S., and Walter Ukovich. (2000). Exact and Heuristic Solutions for a Shipment Problem with Given Frequencies. *Management Science*, *46*(7), 973-988. doi: doi:10.1287/mnsc.46.7.973.12032.

Moin, Noor Hasnah, and Said Salhi. (2007) "Inventory routing problems: a logistical overview." *Journal of the Operational Research Society* 58.9: 1185-1194.

Munkres, James. (1957) "Algorithms for the assignment and transportation problems." *Journal of the Society for Industrial and Applied Mathematics* 5.1: 32-38.

Pochet, Yves, and Laurence A. Wolsey. (2006) Production planning by mixed integer programming. Springer Science & Business Media.

Ronald H. Ballou, "Business logistics/ supply chain management" Pearson Education International 5th Edition.

Samet, Dov, Yair Tauman, and Israel Zang. (1984) "An application of the Aumann-Shapley prices for cost allocation in transportation problems." *Mathematics of Operations Research* 9.1: 25-42.

Sayarshad, H. R., & Ghoseiri, K. (2009). A simulated annealing approach for the multi-periodic rail-car fleet sizing problem. *Computers & Operations Research*, *36*(6), 1789-1799. doi: http://dx.doi.org/10.1016/j.cor.2008.05.004

Scholl, Armin, Robert Klein, and Christian Jürgens. (1997) "BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem."Computers & Operations Research 24.7: 627-645.

Solyalı, O., Süral, H. (2011) "A branch-and-cut algorithm using a strong formulation and a priori tour-based heuristic for an inventory routing problem", Transportation Science, Vol. 45, No. 3, pp. 335-345.

Speranza, M. G., & Ukovich, W. (1994). Minimizing Transportation and Inventory Costs for Several Products on a Single Link. *Operations Research, 42*(5), 879-894. doi: doi:10.1287/opre.42.5.879.

Speranza, M. G., & Ukovich, W. (1996). An algorithm for optimal shipments with given frequencies. *Naval Research Logistics (NRL), 43*(5), 655-671. doi: 10.1002/(SICI)1520-6750(199608)43:5<655::AID-NAV4>3.0.CO;2-4.

Wolsey, L. A. (1998a). *Integer programming*: Wiley.

Wolsey, L. A. (1998b). *Integer programming* (Vol. 42): Wiley New York.

Żak, J., Redmer, A., & Sawicki, P. (2011). Multiple objective optimization of the fleet sizing problem for road freight transportation. *Journal of Advanced Transportation, 45*(4), 321-347. doi: 10.1002/atr.111.

"Bin Packing Problems, (2003)" erişim adresi: http://www.wikipedia.org, Erişim tarihi: 6 Haziran 2014.

APPENDICES

Appendix1: CPLEX Code

Appendix2: JAVA Code

Appendix1

CPLEX Code

```
// problem size
int n=...;// number of customers
int m=...;// number of vehicles
int f=...;// number of frequencies
int d=...;// number of days
int H=...;//number of weeks
int g=...;
int s=5;// number of coinciding frequency sets
int t1 = 17; // number of beginning set members
int t2 = 18;
int t3 = 19;
int t4 = 15;
int t5 = 13;
int t6 = 10;
int t7 = 7;
int t8 = 16;
int t9 = 4;
int t10 = 20;
int t11 = 14;
int t12 = 11;
int t13 = 7;
int t14 = 1;
```

```
 int t15 = 8;
 int t16 = 12;
 int t17 = 20;
 int t18 = 3;


{int} F1={1,2,7,12,17};// Mondays
{int} F2={1,3,8,13,18};//Teusdays
{int} F3={1,4,9,14,19};//Wednesday
{int} F4={1,5,10,15,20};//Thursday
{int} F5={1,6,11,16,21};//Friday
{int} H1=...;// Weekly
{int} H2=...;//Every two weeks
{int} H3=...;//Every three weeks
{int} H4=...;//Every four weeks


 range ccounter=1..n; //customer counter
 range vcounter=1..m; //vehicle counter
 range fcounter=1..f;//frequency counter
 range dcounter=1..d;//day counter
 range Hcounter=1..H;// Week counter
 range Dcounter=1..s;
 range tcounter1 = 1..t1;// counter for end and beginning members
 range tcounter2 = 1..t2;// counter for end and beginning sets
 range tcounter3 = 1..t3;
 range tcounter4 = 1..t4;
 range tcounter5 = 1..t5;
```

```
range tcounter6 = 1..t6;

range tcounter7 = 1..t7;

range tcounter8 = 1..t8;

range tcounter9 = 1..t9;

range tcounter10 = 1..t10;

range tcounter11 = 1..t11;

range tcounter12 = 1..t12;

range tcounter13 = 1..t13;

range tcounter14 = 1..t14;

range tcounter15 = 1..t15;

range tcounter16 = 1..t16;

range tcounter17 = 1..t17;

range tcounter18 = 1..t18;


// Random Cover sets for normal demand

int p1 = 9;

int q1 = 4;

range pcounter1 = 1..p1;

range qcounter1 = 1..q1;

int RndCov7_f2[pcounter1][qcounter1] =...;

int RndCov7_f5[pcounter1][qcounter1] = ...;


int p2 = 10;

range pcounter2 = 1..p2;

int RndCov7_f3[pcounter2][qcounter1] = ...;

int RndCov7_f4[pcounter2][qcounter1] = ...;

int RndCov7_f6[pcounter2][qcounter1] = ...;


int RndCov14_f7[pcounter2][qcounter1] = ...;
```

```
 int RndCov14_f8[pcounter2][qcounter1] = ...;
 int RndCov14_f9[pcounter2][qcounter1] = ...;
 int RndCov14_f10[pcounter2][qcounter1] = ...;
 int RndCov14_f11[pcounter2][qcounter1] = ...;

 int p3 = 32;
 int q2 = 3;
 range pcounter3 = 1..p3;
 range qcounter2 = 1..q2;
 int RndCov7_f7[pcounter3][qcounter2] = ...;

 int p4 = 12;
 range pcounter4 = 1..p4;
 int RndCov7_f8[pcounter4][qcounter2] = ...;

int p5 = 13;
range pcounter5 = 1..p5;
int RndCov7_f9[pcounter5][qcounter2] = ...;

int q3 = 2;
int p6 = 7;
range pcounter6 = 1..p6;
range qcounter3 = 1..q3;
int RndCov7_f10[pcounter6][qcounter3] = ...;

int p7 = 6;
int q4 = 6;
range pcounter7 = 1..p7;
range qcounter4 = 1..q4;
```

```
int RndCov10_f2[pcounter7][qcounter4] = ...;
int RndCov10_f3[pcounter7][qcounter4] = ...;
int RndCov10_f4[pcounter7][qcounter4] = ...;
int RndCov10_f5[pcounter7][qcounter4] = ...;
int RndCov10_f6[pcounter7][qcounter4] = ...;

int RndCov10_f7[pcounter4][qcounter2] = ...;
int RndCov10_f8[pcounter4][qcounter2] = ...;
int RndCov10_f9[pcounter4][qcounter2] = ...;
int RndCov10_f10[pcounter4][qcounter2] = ...;
int RndCov10_f11[pcounter4][qcounter2] = ...;

int RndCov10_f12[pcounter2][qcounter3] = ...;

int p8 = 4;
int q5 = 8;
range pcounter8 = 1..p8;
range qcounter5 = 1..q5;
int RndCov14_f2[pcounter8][qcounter5] = ...;
int RndCov14_f3[pcounter8][qcounter5] = ...;
int RndCov14_f4[pcounter8][qcounter5] = ...;
int RndCov14_f5[pcounter8][qcounter5] = ...;
int RndCov14_f6[pcounter8][qcounter5] = ...;
int RndCov14_f12[pcounter4][qcounter2] = ...;
int RndCov14_f13[pcounter4][qcounter2] = ...;
int RndCov14_f14[pcounter7][qcounter2] = ...;

int p9 = 2;
int q6 = 10;
```

```
int q7 = 11;

int p10 = 11;

int q8 = 14;

int q9 = 7;

int p11 = 8;

int q10 = 5;

int p12 = 22;

int p13 = 1;

range pcounter9 = 1..p9;

range pcounter10 = 1..p10;

range qcounter6 = 1..q6;

range qcounter7 = 1..q7;

range qcounter8 = 1..q8;

range qcounter9 = 1..q9;

range pcounter11 = 1..p11;

range qcounter10 = 1..q10;

range pcounter12 = 1..p12;

range pcounter13 = 1..p13;



int RndCov20_f2[pcounter9][qcounter6] = ...;

int RndCov20_f3[pcounter9][qcounter7] = ...;

int RndCov20_f4[pcounter9][qcounter7] = ...;

int RndCov20_f5[pcounter9][qcounter7] = ...;

int RndCov20_f6[pcounter9][qcounter7] = ...;

int RndCov20_f7[pcounter7][qcounter4] = ...;

int RndCov20_f8[pcounter7][qcounter4] = ...;

int RndCov20_f9[pcounter7][qcounter4] = ...;

int RndCov20_f10[pcounter7][qcounter4] = ...;
```

```
int RndCov20_f11[pcounter7][qcounter4] = ...;
int RndCov20_f12[pcounter2][qcounter1] = ...;
int RndCov20_f13[pcounter2][qcounter1] = ...;
int RndCov20_f14[pcounter2][qcounter1] = ...;
int RndCov20_f15[pcounter2][qcounter1] = ...;
int RndCov20_f16[pcounter2][qcounter1] = ...;
int RndCov20_f17[pcounter6][qcounter2] = ...;

int RndCov21_f2[pcounter9][qcounter7] = ...;
int RndCov21_f3[pcounter9][qcounter7] = ...;
int RndCov21_f4[pcounter9][qcounter7] = ...;
int RndCov21_f5[pcounter9][qcounter7] = ...;
int RndCov21_f6[pcounter9][qcounter7] = ...;
int RndCov21_f7[pcounter7][qcounter4] = ...;
int RndCov21_f8[pcounter7][qcounter4] = ...;
int RndCov21_f9[pcounter7][qcounter4] = ...;
int RndCov21_f10[pcounter7][qcounter4] = ...;
int RndCov21_f11[pcounter7][qcounter4] = ...;
int RndCov21_f12[pcounter2][qcounter1] = ...;
int RndCov21_f13[pcounter2][qcounter1] = ...;
int RndCov21_f14[pcounter2][qcounter1] = ...;
int RndCov21_f15[pcounter1][qcounter1] = ...;
int RndCov21_f16[pcounter2][qcounter1] = ...;
int RndCov21_f17[pcounter10][qcounter2] = ...;

int RndCov27_f2[pcounter9][qcounter8] = ...;
int RndCov27_f3[pcounter9][qcounter8] = ...;
int RndCov27_f4[pcounter9][qcounter8] = ...;
int RndCov27_f5[pcounter9][qcounter8] = ...;
```

```
int RndCov27_f6[pcounter9][qcounter8] = ...;

int RndCov27_f7[pcounter8][qcounter5] = ...;

int RndCov27_f8[pcounter8][qcounter5] = ...;

int RndCov27_f9[pcounter8][qcounter9] = ...;

int RndCov27_f10[pcounter8][qcounter5] = ...;

int RndCov27_f11[pcounter8][qcounter5] = ...;

int RndCov27_f12[pcounter11][qcounter10] = ...;

int RndCov27_f13[pcounter12][qcounter6] = ...;

int RndCov27_f14[pcounter6][qcounter4] = ...;

int RndCov27_f15[pcounter11][qcounter10] = ...;

int RndCov27_f16[pcounter11][qcounter10] = ...;

int RndCov27_f17[pcounter2][qcounter1] = ...;

int RndCov27_f18[pcounter2][qcounter1] = ...;

int RndCov27_f19[pcounter2][qcounter1] = ...;

int RndCov27_f20[pcounter13][qcounter1] = ...;


/* Random cut sets for 50D*/

int q11 = 17;

int p14 = 18;

int p15 = 17;

int p16 = 20;

int p17 = 14;

int p18 = 15;

int p19 = 3;

int p20 = 5;

int q12 = 9;

range qcounter11 = 1..q11;

range pcounter14 = 1..p14;

range pcounter15 = 1..p15;
```

```
range pcounter16 = 1..p16;

range pcounter17 = 1..p17;

range pcounter18 = 1..p18;

range pcounter19 = 1..p19;

range pcounter20 = 1..p20;

int RndCov7_f1_50[pcounter9][qcounter11] = ...;

int RndCov7_f2_50[pcounter10][qcounter2] = ...;

int RndCov7_f3_50[pcounter4][qcounter2] = ...;

int RndCov7_f4_50[pcounter10][qcounter2] = ...;

int RndCov7_f5_50[pcounter4][qcounter2] = ...;

int RndCov7_f6_50[pcounter10][qcounter2] = ...;

int RndCov7_f7_50[pcounter14][qcounter2] = ...;

int RndCov7_f8_50[pcounter15][qcounter3] = ...;

int RndCov7_f9_50[pcounter8][qcounter3] = ...;


int RndCov10_f2_50[pcounter2][qcounter1] = ...;

int RndCov10_f3_50[pcounter2][qcounter1] = ...;

int RndCov10_f4_50[pcounter2][qcounter1] = ...;

int RndCov10_f5_50[pcounter1][qcounter1] = ...;

int RndCov10_f6_50[pcounter2][qcounter1] = ...;

int RndCov10_f7_50[pcounter16][qcounter3] = ...;

int RndCov10_f8_50[pcounter17][qcounter3] = ...;

int RndCov10_f9_50[pcounter18][qcounter3] = ...;

int RndCov10_f10_50[pcounter9][qcounter3] = ...;


int RndCov14_f2_50[pcounter11][qcounter10] = ...;

int RndCov14_f3_50[pcounter11][qcounter10] = ...;

int RndCov14_f5_50[pcounter11][qcounter10] = ...;

int RndCov14_f4_50[pcounter6][qcounter4] = ...;
```

```
int RndCov14_f6_50[pcounter6][qcounter4] = ...;
int RndCov14_f7_50[pcounter4][qcounter2] = ...;
int RndCov14_f8_50[pcounter4][qcounter2] = ...;
int RndCov14_f9_50[pcounter4][qcounter2] = ...;
int RndCov14_f10_50[pcounter4][qcounter2] = ...;
int RndCov14_f11_50[pcounter10][qcounter2] = ...;
int RndCov14_f12_50[pcounter19][qcounter3] = ...;

int RndCov20_f2_50[pcounter8][qcounter9] = ...;
int RndCov20_f3_50[pcounter8][qcounter9] = ...;
int RndCov20_f4_50[pcounter8][qcounter5] = ...;
int RndCov20_f5_50[pcounter8][qcounter5] = ...;
int RndCov20_f6_50[pcounter8][qcounter5] = ...;
int RndCov20_f7_50[pcounter2][qcounter1] = ...;
int RndCov20_f8_50[pcounter2][qcounter1] = ...;
int RndCov20_f9_50[pcounter2][qcounter1] = ...;
int RndCov20_f10_50[pcounter2][qcounter1] = ...;
int RndCov20_f11_50[pcounter2][qcounter1] = ...;
int RndCov20_f12_50[pcounter4][qcounter2] = ...;
int RndCov20_f13_50[pcounter4][qcounter2] = ...;
int RndCov20_f14_50[pcounter20][qcounter2] = ...;

int RndCov21_f2_50[pcounter8][qcounter5] = ...;
int RndCov21_f3_50[pcounter8][qcounter5] = ...;
int RndCov21_f4_50[pcounter8][qcounter5] = ...;
int RndCov21_f5_50[pcounter8][qcounter5] = ...;
int RndCov21_f6_50[pcounter8][qcounter5] = ...;
int RndCov21_f7_50[pcounter1][qcounter1] = ...;
int RndCov21_f8_50[pcounter2][qcounter1] = ...;
```

```
int RndCov21_f9_50[pcounter2][qcounter1] = ...;
int RndCov21_f10_50[pcounter2][qcounter1] = ...;
int RndCov21_f11_50[pcounter2][qcounter1] = ...;
int RndCov21_f12_50[pcounter4][qcounter2] = ...;
int RndCov21_f13_50[pcounter4][qcounter2] = ...;
int RndCov21_f14_50[pcounter6][qcounter2] = ...;

int RndCov27_f2_50[pcounter8][qcounter6] = ...;
int RndCov27_f3_50[pcounter8][qcounter6] = ...;
int RndCov27_f4_50[pcounter8][qcounter6] = ...;
int RndCov27_f5_50[pcounter8][qcounter6] = ...;
int RndCov27_f6_50[pcounter8][qcounter6] = ...;
int RndCov27_f7_50[pcounter11][qcounter10] = ...;
int RndCov27_f8_50[pcounter11][qcounter10] = ...;
int RndCov27_f9_50[pcounter11][qcounter10] = ...;
int RndCov27_f10_50[pcounter11][qcounter10] = ...;
int RndCov27_f11_50[pcounter11][qcounter10] = ...;
int RndCov27_f12_50[pcounter2][qcounter1] = ...;
int RndCov27_f13_50[pcounter10][qcounter1] = ...;
int RndCov27_f14_50[pcounter10][qcounter1] = ...;
int RndCov27_f15_50[pcounter1][qcounter1] = ...;


 //parameters
 float r[vcounter]=...;// approximate cost of a route
 int cc=17; // constant cost of a route
 int A[vcounter]=...;// vehicle renting cost
 float Y[ccounter][fcounter]=...;// Customer demand
 int h=300; // inventory holding cost
```

```
float k[ccounter][fcounter]=...;// ordering cost

int S=5; // maximum number of cusomers to visit in a day

int cap[vcounter]=...;// capacity of vehicles

int M=21;// big number

int F[fcounter]=...;// frequency dependent repetition number

int a[ccounter][ccounter]=...;// incidence matrice of customers

int W[vcounter][fcounter]=...;// normal frequency inequality

int V[vcounter][Dcounter]=...;// coinciding frequencies inequality

int End1[tcounter1] = ...; // end set for cover1

int End2[tcounter2] = ... ;// end set for cover2

int End3[tcounter3] = ... ;

int End4[tcounter4] = ...;

int End5[tcounter5] = ...;

int End6[tcounter6] = ...;

int End6_1[tcounter6] = ...;

int End7[tcounter7] = ...;

int End5_1[tcounter5] = ...;

int End8[tcounter8] = ...;

int End9[tcounter9] = ...;

int End10[tcounter10]=...;

int End3_1[tcounter3] = ...;

int End11[tcounter11]= ...;

int End12[tcounter12]= ...;

int Ende10[tcounter10] = ...;

int Ende2[tcounter2] = ...;

int Ende6[tcounter6] = ...;

int Ende13[tcounter13] = ...;

int Ende14[tcounter14] = ...;

int Ende6p[tcounter6] = ...;
```

```
int Ende15[tcounter15]= ...;
int Ende16[tcounter16] = ...;
int Ende11[tcounter11] = ...;
int Ende17[tcounter17] = ...;
int Ende11p[tcounter11] = ...;
int Ende2p[tcounter2] = ...;
int Ende10p[tcounter10] = ...;
int End16[tcounter16] = ...;
int Ende15p[tcounter15] = ...;
int Ende8[tcounter8] = ...;
int Ende13p[tcounter13] = ...;
int Ende2pr[tcounter2] = ...;
int Ende18[tcounter18] = ...;
int Ende11pr[tcounter11] = ...;
```

```
// decision variables
 dvar boolean x[ccounter][vcounter][fcounter];
dvar boolean CC[vcounter][fcounter];
dvar int+ c[vcounter][fcounter];
dvar boolean CV[vcounter];
dvar float AvLotSize;
dvar float AvFrequency;
dvar boolean R[dcounter][vcounter][Hcounter];
dvar float Z;
dvar float P;
dvar float RoutingCost;
dvar float RentingCost;
```

```
dvar float Inventory1;

dvar float Inventory2;

dvar float TotalLot[vcounter];

dvar int NumCustomer[vcounter];


//model

minimize sum(v in vcounter) F[1]*CC[v][1]*cc+ sum(d in dcounter,H in
Hcounter,v in vcounter)R[d][v][H]* cc +sum(f in fcounter,v in
vcounter)c[v][f]*r[v]*F[f]+sum(v in vcounter)A[v]*CV[v]+sum(i in
ccounter,v in vcounter,f in fcounter)(x[i][v][f]*Y[i][f])*1/2*h+sum(i in
ccounter,v in vcounter,f in fcounter)x[i][v][f]*k[i][f];


subject to{


     forall(i in ccounter){

       constallocation:

       sum(v in vcounter,f in fcounter) x[i][v][f]==1;// demand

    }

    forall(f in fcounter, v in vcounter){

     cut1:

     sum(i in ccounter) x[i][v][f]<= W[v][f]-1;

    }

    forall(v in vcounter){

       cut2:

       sum(i in ccounter, f in F1) x[i][v][f]<= V[v][1] - 1;

       sum(i in ccounter, f in F2) x[i][v][f]<= V[v][2] - 1;

       sum(i in ccounter, f in F3) x[i][v][f]<= V[v][3] - 1;

       sum(i in ccounter, f in F4) x[i][v][f]<= V[v][4] - 1;

       sum(i in ccounter, f in F5) x[i][v][f]<= V[v][5] - 1;

    }

  forall(v in vcounter,f in fcounter)
```

```
sum(i in ccounter)x[i][v][f]==c[v][f];// number of customers


        forall(v in vcounter,f in fcounter ){

         constroute:

         CC[v][f]*S>= c[v][f];//route existance

         CC[v][f]<=c[v][f];

         }

            forall(v in vcounter)

              CV[v]*M>=sum(f in fcounter ,i in
ccounter)x[i][v][f];//vehicle number

           forall(v in vcounter,f in fcounter )

            capconst:

            sum(i in ccounter)((x[i][v][f]*Y[i][f]))<=cap[v];

             forall(v in vcounter){

              maximumcustomer:

               sum(f in F1) c[v][f]<=S;

               sum(f in F2) c[v][f]<=S;

               sum(f in F3) c[v][f]<=S;

               sum(f in F4) c[v][f]<=S;

               sum(f in F5) c[v][f]<=S;

              }


               forall(v in vcounter){

                vnumber1:

                 sum(i in ccounter,f in
F1)(x[i][v][f]*Y[i][f])<=cap[v];

                 sum(i in ccounter,f in
F2)(x[i][v][f]*Y[i][f])<=cap[v];

                 sum(i in ccounter,f in
F3)(x[i][v][f]*Y[i][f])<=cap[v];
```

```
                        sum(i in ccounter,f in
F4)(x[i][v][f]*Y[i][f])<=cap[v];

                        sum(i in ccounter,f in
F5)(x[i][v][f]*Y[i][f])<=cap[v];

                    }

                    forall(v in vcounter,i in ccounter,j in ccounter: i<j){

                    sum(f in F1)x[i][v][f] + sum(f in F1)x[j][v][f]<=
a[i][j];

                    sum(f in F2)x[i][v][f] + sum(f in F2)x[j][v][f]<=
a[i][j];

                    sum(f in F3)x[i][v][f] + sum(f in F3)x[j][v][f]<=
a[i][j];

                    sum(f in F4)x[i][v][f] + sum(f in F4)x[j][v][f]<=
a[i][j];

                    sum(f in F5)x[i][v][f] + sum(f in F5)x[j][v][f]<=
a[i][j];

                    }
                      forall(v in vcounter,H in H1)
                       R[1][v][H]>= CC[v][2]- CC[v][1];
                       forall(v in vcounter,H in H2)
                       R[1][v][H]>= CC[v][7]- CC[v][1];
                       forall(v in vcounter,H in H3)
                       R[1][v][H]>= CC[v][12]- CC[v][1];
                       forall(v in vcounter,H in H4)
                       R[1][v][H]>= CC[v][17]- CC[v][1];


                        forall(v in vcounter,H in H1)
                       R[2][v][H]>= CC[v][3]- CC[v][1];
                       forall(v in vcounter,H in H2)
                       R[2][v][H]>= CC[v][8]- CC[v][1];
                       forall(v in vcounter,H in H3)
                       R[2][v][H]>= CC[v][13]- CC[v][1];
```

```
forall(v in vcounter,H in H4)
R[2][v][H]>= CC[v][18]- CC[v][1];

forall(v in vcounter,H in H1)
R[3][v][H]>= CC[v][4]- CC[v][1];
forall(v in vcounter,H in H2)
R[3][v][H]>= CC[v][9]- CC[v][1];
forall(v in vcounter,H in H3)
R[3][v][H]>= CC[v][14]- CC[v][1];
forall(v in vcounter,H in H4)
R[3][v][H]>= CC[v][19]- CC[v][1];

 forall(v in vcounter,H in H1)
R[4][v][H]>= CC[v][5]- CC[v][1];
forall(v in vcounter,H in H2)
R[4][v][H]>= CC[v][10]- CC[v][1];
forall(v in vcounter,H in H3)
R[4][v][H]>= CC[v][15]- CC[v][1];
forall(v in vcounter,H in H4)
R[4][v][H]>= CC[v][20]- CC[v][1];

 forall(v in vcounter,H in H1)
R[5][v][H]>= CC[v][6]- CC[v][1];
forall(v in vcounter,H in H2)
R[5][v][H]>= CC[v][11]- CC[v][1];
forall(v in vcounter,H in H3)
R[5][v][H]>= CC[v][16]- CC[v][1];
forall(v in vcounter,H in H4)
R[5][v][H]>= CC[v][21]- CC[v][1];
```

```
      AvLotSize==     sum(i     in     ccounter,v     in     vcounter,f     in
fcounter)(x[i][v][f]*Y[i][f])/n;

                  AvFrequency==sum(i   in   ccounter,v   in   vcounter,f   in
fcounter)(x[i][v][f]*f)/n;

                   Z==    sum(v  in  vcounter)  F[1]*CC[v][1]*cc+  sum(d  in
dcounter,H   in   Hcounter,v   in   vcounter)R[d][v][H]*   cc   +sum(v   in
vcounter)A[v]*CV[v]+sum(f in fcounter,v in vcounter)c[v][f]*r[v]*F[f];

                   P==sum(i     in     ccounter,v     in     vcounter,f     in
fcounter)(x[i][v][f]*Y[i][f])*1/2*h+sum(i  in  ccounter,v  in  vcounter,f  in
fcounter)x[i][v][f]*k[i][f];

                   RoutingCost==  sum(v  in  vcounter)F[1]*CC[v][1]*cc+sum(d
in dcounter,H in Hcounter,v in vcounter)R[d][v][H]* cc+sum(f in fcounter,v
in vcounter)c[v][f]*r[v]*F[f];

                  RentingCost==sum(v in vcounter)A[v]*CV[v];

                   Inventory1==  sum(i  in  ccounter,v  in  vcounter,f  in
fcounter)(x[i][v][f]*Y[i][f])*1/2*h;

                   Inventory2==  sum(i  in  ccounter,v  in  vcounter,f  in
fcounter)x[i][v][f]*k[i][f];

                   forall(v in vcounter){

                   TotalLot[v]==sum(f   in   fcounter,i   in   ccounter   )
x[i][v][f]*Y[i][f];

                    NumCustomer[v]==sum(f in fcounter)c[v][f];

                  }



                  /* Valid inequalities for Normal demand scenarios*/

     // cap 7, weekly

                  VI1_cap7:

                  forall(v  in  vcounter,  f  in  fcounter,  t  in  tcounter1  :
3<=v<=4 && 2<=f<=6)

                    sum(i  in  ccounter  :  t<=  i  <=  End1[t])  x[i][v][f]  <=
End1[t] - t;




                 // cap 7 , bi-weekly
```

```
                VI2_cap7:

               forall(v in vcounter, f in fcounter, t in tcounter2:
3<=v<=4 && 7<=f<=11)

                sum(i in ccounter : t<= i<= End2[t]) x[i][v][f] <=
End2[t] - t;


                // cap 7, thrice-weekly

                VI3_cap7:

                forall(v in vcounter, f in fcounter, t in tcounter3 :
3<=v<=4 && 12<=f<=16)

                   sum(i in ccounter : t<= i<= End3[t]) x[i][v][f] <=
End3[t] - t;


               // cap 7, quad-weekly

               VI4_cap7:

               forall(v in vcounter, f in fcounter, t in tcounter3 :
3<=v<=4 && 17<=f<=21)

                sum(i in ccounter : t<= i<= End3[t]) x[i][v][f] <= End3[t]
- t;



              // cap 10, weekly

              VI1_cap10:

               forall(v in vcounter, f in fcounter, t in tcounter4 :
1<=v<=2 && 2<=f<=6)

                   sum(i in ccounter : t<= i<= End4[t]) x[i][v][f] <=
End4[t] - t;


             // cap 10, bi-weekly

             VI2_cap10:

                forall(v in vcounter, f in fcounter, t in tcounter2 :
1<=v<=2 && 7<=f<=11)

                   sum(i in ccounter : t<= i<= End2[t]) x[i][v][f] <=
End2[t] - t;
```

```
            // cap 10, thrice-weekly

            VI3_cap10:

              forall(v in vcounter, f in fcounter, t in tcounter3 :
1<=v<=2 && 12<=f<=16)

                    sum(i in ccounter : t<= i<= End3[t]) x[i][v][f] <=
End3[t] - t;


           // cap 10, quad-weekly

           VI4_cap10:

              forall(v in vcounter, f in fcounter, t in tcounter3 :
1<=v<=2 && 17<=f<=21)

                    sum(i in ccounter : t<= i<= End3[t]) x[i][v][f] <=
End3[t] - t;


          // cap 14, weekly

          VI1_cap14:

            forall(v in vcounter, f in fcounter, t in tcounter5: 3<=t<=4
&& 2<=f<=6)

              sum(i in ccounter : t<=i<= End5[t]) x[i][v][f] <= End5[t] -
t;


         // cap 14, bi-weekly

         VI2_cap14:

           forall(v in vcounter, f in fcounter, t in tcounter1 : 3<=v<=4
&& 7<=f<=11)

             sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] -
t;


        // cap 14, thrice-weekly

        VI3_cap14:

          forall(v in vcounter, f in fcounter, t in tcounter2 : 3<=v<=4 &&
12<=f<=16)
```

```
            sum(i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


       // cap 14, quad-weekly

       VI4_cap14:

         forall(v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
17<=f<=21)

           sum(i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


      // cap 20, weekly

      VI1_cap20 :

        forall( v in vcounter, f in fcounter, t in tcounter6: 1<=v<=2 &&
2<=f<=6)

          sum(i in ccounter : t<=i<=End6[6]) x[i][v][f] <= End6[t] - t;


     // cap 20, bi-weekly

     VI2_cap20:

       forall( v in vcounter, f in fcounter, t in tcounter4 : 1<=v<=2 &&
7<=f<=11)

         sum( i in ccounter : t<=i<=End4[t]) x[i][v][f] <= End4[t] - t;


    // cap 20, thrice-weekly

    VI3_cap20:

      forall( v in vcounter, f in fcounter, t in tcounter1: 1<=v<=2 &&
12<=f<=16)

        sum(i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;


   // cap 20, quad_weekly

   VI4_cap20:

     forall( v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<= i<= End2[t]) x[i][v][f] <= End2[t] - t;
```

```
   // cap 21, weekly

   VI1_cap21:

     forall( v in vcounter, f in fcounter, t in tcounter6: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End6_1[t]) x[i][v][f] <= End6_1[t] - t;


  // cap 21, bi-weekly

  VI2_cap21:

    forall(   v in vcounter, f in fcounter, t in tcounter4: 3<=v<=4 &&
7<=f<=11 )

       sum( i in ccounter : t<=i<=End4[t]) x[i][v][f] <= End4[t] - t;


  // cap 21, thrice_weekly

  VI3_cap21:

     forall(   v in vcounter, f in fcounter, t in tcounter1: 3<=v<=4 &&
12<=f<=16 )

       sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;


   // cap 21, quad_weekly

  VI4_cap21:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
17<=f<=21 )

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


  // cap 27,weekly

  VI1_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter7: 1<=v<=2 &&
2<=f<=6 )

       sum( i in ccounter : t<=i<=End7[t]) x[i][v][f] <= End7[t] - t;
```

```
   // cap 27,bi-weekly

  VI2_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter5: 1<=v<=2 &&
7<=f<=11 )

       sum( i in ccounter : t<=i<=End5_1[t]) x[i][v][f] <= End5_1[t] - t;


   // cap 27,thrice-weekly

  VI3_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 1<=v<=2 &&
12<=f<=16 )

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <= End8[t] - t;


  // cap 27,quad-weekly

  VI4_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter1: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;



       /* valid inequalities for 50% demand */


    // cap 7, daily

    VI1_cap7_50:

    forall(v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 && f ==
1)

      sum( i in ccounter: t<=i<=End8[t]) x[i][v][f] <= End8[t] - t;


    // cap 7,weekly

    VI2_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
2<=f<=6)
```

```
       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


   // cap 7,bi-weekly

    VI3_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;


   // cap 7,thrice-weekly

    VI4_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter10: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End10[t]) x[i][v][f] <= End10[t] - t;


  // cap 7,quad-weekly

    VI5_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter10: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End10[t]) x[i][v][f] <= End10[t] - t;


  // cap 10,weekly

    VI1_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter1: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;


  // cap 10,bi-weekly

    VI2_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;
```

```
  // cap 10,thrice-weekly

    VI3_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;


  // cap 10,quad-weekly

    VI4_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;


  // cap 14,weekly

    VI1_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <= End8[t] - t;




  // cap 14,bi-weekly

    VI2_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


  // cap 14,thrice-weekly

    VI3_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;
```

```
  // cap 14,quad-weekly

    VI4_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;


  // cap 20,weekly

    VI1_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End11[t]) x[i][v][f] <= End11[t] - t;


  // cap 20,bi-weekly

    VI2_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter1: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;


  // cap 20,thrice-weekly

    VI3_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


  // cap 20,quad-weekly

    VI4_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter3: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End3[t]) x[i][v][f] <= End3[t] - t;
```

```
  // cap 21,weekly

    VI1_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter5: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End5[t]) x[i][v][f] <= End5[t] - t;


  // cap 21,bi-weekly

    VI2_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter1: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;


  // cap 21,thrice-weekly

    VI3_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


  // cap 21,quad-weekly

    VI4_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;


  // cap 27,weekly

    VI1_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End11[t]) x[i][v][f] <= End11[t] - t;


  // cap 27,bi-weekly
```

```
    VI2_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <= End8[t] - t;


  // cap 27,thrice-weekly

    VI3_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter1: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End1[t]) x[i][v][f] <= End1[t] - t;


  // cap 27,quad-weekly

    VI4_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <= End2[t] - t;



   /*Extra cuts with double increement D*/


    // cap 7,weekly

    VIE1_cap7:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End11[t] && i == i+2) x[i][v][f] <=(
End11[t] - t)/2;


   // cap 7,bi-weekly

    VIE2_cap7:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
7<=f<=11)
```

```
       sum( i in ccounter : t<=i<=Ende2[t]) x[i][v][f] <=( Ende2[t] -
t)/2;


  // cap 7,thrice-weekly

    VIE3_cap7:

     forall(  v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;




     // cap 7,quad-weekly

    VIE4_cap7:

     forall(  v in vcounter, f in fcounter, t in tcounter10: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=Ende10[t]) x[i][v][f] <=( Ende10[t] -
t)/2;



     // cap 10,weekly

    VIE1_cap10:

     forall(  v in vcounter, f in fcounter, t in tcounter6: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende6[t]) x[i][v][f] <=( Ende6[t] -
t)/2;



     // cap 10,bi-weekly

    VIE2_cap10:

     forall(  v in vcounter, f in fcounter, t in tcounter8: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;



     // cap 10,thrice-weekly

    VIE3_cap10:
```

```
     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;


     // cap 10,quad-weekly

    VIE4_cap10:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;


     // cap 14,weekly

    VIE1_cap14:

     forall(   v in vcounter, f in fcounter, t in tcounter13: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende13[t]) x[i][v][f] <=( Ende13[t] -
t)/2;


     // cap 14,bi-weekly

    VIE2_cap14:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End11[t]) x[i][v][f] <=( End11[t] -
t)/2;


    // cap 14,thrice-weekly

    VIE3_cap14:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;


     // cap 14,quad-weekly

    VIE4_cap14:
```

```
     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;


      // cap 20,weekly

    VIE1_cap20:

     forall(   v in vcounter, f in fcounter, t in tcounter14: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende14[t]) x[i][v][f] <=( Ende14[t] -
t)/2;


     // cap 20,bi-weekly

    VIE2_cap20:

     forall(   v in vcounter, f in fcounter, t in tcounter6: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=Ende6[t]) x[i][v][f] <=( Ende6[t] -
t)/2;


   // cap 20,thrice-weekly

    VIE3_cap20:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End11[t]) x[i][v][f] <=( End11[t] -
t)/2;


  // cap 20,thrice-weekly

    VIE4_cap20:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;


  // cap 21,bi-weekly

   VIE1_cap21:
```

```
     forall(   v in vcounter, f in fcounter, t in tcounter6: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=Ende6p[t]) x[i][v][f] <=( Ende6p[t] -
t)/2;



    // cap 21,thrice-weekly

    VIE2_cap21:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End11[t]) x[i][v][f] <=( End11[t] -
t)/2;



    // cap 21,quad-weekly

    VIE3_cap21:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;



     // cap 27,bi-weekly

    VIE1_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter15: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=Ende15[t]) x[i][v][f] <=( Ende15[t] -
t)/2;



    // cap 27,thrice-weekly

    VIE2_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter16: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=Ende16[t]) x[i][v][f] <=( Ende16[t] -
t)/2;



     // cap 27,quad-weekly
```

```
    VIE3_cap27:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=Ende11[t]) x[i][v][f] <=( Ende11[t] -
t)/2;



       /*Extra cuts with double increement 50D*/


    // cap 7,weekly

    VIE1_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;


    // cap 7,bi-weekly

    VIE2_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;


   // cap 7,thrice-weekly

    VIE3_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter17: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=Ende17[t]) x[i][v][f] <=( Ende17[t] -
t)/2;


    // cap 7,quad-weekly

    VIE4_cap7_50:

     forall(   v in vcounter, f in fcounter, t in tcounter17: 3<=v<=4 &&
17<=f<=21)
```

```
       sum( i in ccounter : t<=i<=Ende17[t]) x[i][v][f] <=( Ende17[t] -
t)/2;


      // cap 10,weekly

    VIE1_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende11p[t]) x[i][v][f] <=( Ende11p[t] -
t)/2;


    // cap 10,bi-weekly

    VIE2_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;




    // cap 10,thrice-weekly

    VIE3_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=Ende2p[t]) x[i][v][f] <=( Ende2p[t] -
t)/2;


   // cap 10,quad-weekly

    VIE4_cap10_50:

     forall(   v in vcounter, f in fcounter, t in tcounter10: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=Ende10p[t]) x[i][v][f] <=( Ende10p[t] -
t)/2;


    // cap 14,weekly

    VIE1_cap14_50:
```

```
     forall(   v in vcounter, f in fcounter, t in tcounter16: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=End16[t]) x[i][v][f] <=( End16[t] -
t)/2;



    // cap 14,bi-weekly

    VIE2_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;



    // cap 14,thrice-weekly

    VIE3_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;



     // cap 14,quad-weekly

    VIE4_cap14_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;



    // cap 20,weekly

    VIE1_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter15: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende15p[t]) x[i][v][f] <=( Ende15p[t] -
t)/2;



    // cap 20,bi-weekly

    VIE2_cap20_50:
```

```
     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=Ende11p[t]) x[i][v][f] <=( Ende11p[t] -
t)/2;



    // cap 20,thrice-weekly

    VIE3_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=Ende8[t]) x[i][v][f] <=( Ende8[t] -
t)/2;



    // cap 20,quad-weekly

    VIE4_cap20_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=End2[t]) x[i][v][f] <=( End2[t] - t)/2;



     // cap 21,weekly

    VIE1_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter13: 3<=v<=4 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende13p[t]) x[i][v][f] <=( Ende13p[t] -
t)/2;



    // cap 21,bi-weekly

    VIE2_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 3<=v<=4 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=End11[t]) x[i][v][f] <=( End11[t] -
t)/2;



     // cap 21,thrice-weekly
```

```
    VIE3_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 3<=v<=4 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=End8[t]) x[i][v][f] <=( End8[t] - t)/2;


    // cap 21,quad-weekly

    VIE4_cap21_50:

     forall(   v in vcounter, f in fcounter, t in tcounter2: 3<=v<=4 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=Ende2pr[t]) x[i][v][f] <=( Ende2pr[t] -
t)/2;


   // cap 27,weekly

    VIE1_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter18: 1<=v<=2 &&
2<=f<=6)

       sum( i in ccounter : t<=i<=Ende18[t]) x[i][v][f] <=( Ende18[t] -
t)/2;


   // cap 27,bi-weekly

    VIE2_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter16: 1<=v<=2 &&
7<=f<=11)

       sum( i in ccounter : t<=i<=Ende16[t]) x[i][v][f] <=( Ende16[t] -
t)/2;


     // cap 27,thrice-weekly

    VIE3_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter11: 1<=v<=2 &&
12<=f<=16)

       sum( i in ccounter : t<=i<=Ende11pr[t]) x[i][v][f] <=( Ende11pr[t]
- t)/2;
```

```
    // cap 27,quad-weekly

    VIE4_cap27_50:

     forall(   v in vcounter, f in fcounter, t in tcounter8: 1<=v<=2 &&
17<=f<=21)

       sum( i in ccounter : t<=i<=Ende8[t]) x[i][v][f] <=( Ende8[t] -
t)/2;



       /* Rnadom covers, D*/


       //cap7

       forall( v in vcounter, f in fcounter, p in pcounter1 : f ==2 &&
3<=v<=4)

         sum(i in ccounter, q in qcounter1 : i == RndCov7_f2[p][q])
x[i][v][f] <= 3;


       forall( v in vcounter, f in fcounter, p in pcounter2 : f ==3 &&
3<=v<=4)

         sum(i in ccounter, q in qcounter1 : i == RndCov7_f3[p][q])
x[i][v][f] <= 3;


       forall( v in vcounter, f in fcounter, p in pcounter2 : f ==4 &&
3<=v<=4)

         sum(i in ccounter, q in qcounter1 : i == RndCov7_f4[p][q])
x[i][v][f] <= 3;


       forall( v in vcounter, f in fcounter, p in pcounter1 : f ==5 &&
3<=v<=4)

         sum(i in ccounter, q in qcounter1 : i == RndCov7_f5[p][q])
x[i][v][f] <= 3;


       forall( v in vcounter, f in fcounter, p in pcounter2 : f ==6 &&
3<=v<=4)

         sum(i in ccounter, q in qcounter1 : i == RndCov7_f6[p][q])
x[i][v][f] <= 3;
```

```
       forall( v in vcounter, f in fcounter, p in pcounter3 : f ==7 &&
3<=v<=4 &&  p!= 31 )

         sum(i in ccounter, q in qcounter2 : i == RndCov7_f7[p][q])
x[i][v][f] <= 1;

       forall( v in vcounter, f in fcounter, p in pcounter3 : f ==7 &&
3<=v<=4 && p== 31)

         sum(i in ccounter, q in qcounter2 : i == RndCov7_f7[p][q])
x[i][v][f] <= 2;



       forall( v in vcounter, f in fcounter, p in pcounter4 : f ==8 &&
3<=v<=4 && p != 11)

         sum(i in ccounter, q in qcounter2 : i == RndCov7_f8[p][q])
x[i][v][f] <= 1;

       forall( v in vcounter, f in fcounter, p in pcounter4 : f ==8 &&
3<=v<=4 && p4== 11 )

         sum(i in ccounter, q in pcounter2 : i == RndCov7_f8[p][q])
x[i][v][f] <= 2;



       forall( v in vcounter, f in fcounter, p in pcounter5 : f ==9 &&
3<=v<=4 && p!= 3 && p != 13)

         sum(i in ccounter, q in qcounter2 : i == RndCov7_f9[p][q])
x[i][v][f] <= 1;

       forall( v in vcounter, f in fcounter, p in pcounter5 : f ==9 &&
3<=v<=4 && p == 3 && p ==13)

         sum(i in ccounter, q in qcounter2 : i == RndCov7_f9[p][q])
x[i][v][f] <= 2;



       forall( v in vcounter, f in fcounter, p in pcounter6 : f ==10 &&
3<=v<=4)

         sum(i in ccounter,q in qcounter3 : i == RndCov7_f10[p][q])
x[i][v][f] <= 1;



         //cap10
```

```
       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==2 &&
1<=v<=2 && p != 1 )

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f2[p][q])
x[i][v][f] <= 5;

       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==2 &&
1<=v<=2 && p == 1 )

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f2[p][q])
x[i][v][f] <= 4;



       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==3 &&
1<=v<=2 && p!= 1)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f3[p][q])
x[i][v][f] <= 5;

       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==3 &&
1<=v<=2 && p== 1)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f3[p][q])
x[i][v][f] <= 4;



       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==4 &&
1<=v<=2 && p!= 3 && p!=5 && p!=6)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f4[p][q])
x[i][v][f] <= 5;

       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==4 &&
1<=v<=2 && p== 3 && p==5 && p==6)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f4[p][q])
x[i][v][f] <= 4;



       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==5 &&
1<=v<=2 && p!= 1 && p!=3 && p!=6)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f5[p][q])
x[i][v][f] <= 5;

       forall( v in vcounter, f in fcounter, p in pcounter7 : f ==5 &&
1<=v<=2 && p== 1 && p==3 && p==6)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f5[p][q])
x[i][v][f] <= 4;
```

```
      forall( v in vcounter, f in fcounter, p in pcounter7 : f ==6 &&
1<=v<=2 && p!= 1 && p!=3 && p!=6 && p!=5)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f6[p][q])
x[i][v][f] <= 5;

      forall( v in vcounter, f in fcounter, p in pcounter7 : f ==6 &&
1<=v<=2 && p== 1 && p==3 && p==6 && p==5)

         sum(i in ccounter, q in qcounter4 : i == RndCov10_f6[p][q])
x[i][v][f] <= 4;



      forall( v in vcounter, f in fcounter, p in pcounter4 : f ==7 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter2 : i == RndCov10_f7[p][q])
x[i][v][f] <= 2;



      forall( v in vcounter, f in fcounter, p in pcounter4 : f ==8 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter2 : i == RndCov10_f8[p][q])
x[i][v][f] <= 2;



      forall( v in vcounter, f in fcounter, p in pcounter4 : f ==9 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter2 : i == RndCov10_f9[p][q])
x[i][v][f] <= 2;



      forall( v in vcounter, f in fcounter, p in pcounter4 : f ==10 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter2 : i == RndCov10_f10[p][q])
x[i][v][f] <= 2;



      forall( v in vcounter, f in fcounter, p in pcounter4 : f ==11 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter2 : i == RndCov10_f11[p][q])
x[i][v][f] <= 2;



      forall( v in vcounter, f in fcounter, p in pcounter2 : f ==12 &&
1<=v<=2)
```

```
        sum(i in ccounter,q in qcounter3 : i == RndCov10_f12[p][q])
x[i][v][f] <= 1;


    //cap14


     forall( v in vcounter, f in fcounter, p in pcounter8 : f ==2 &&
3<=v<=4 && p!= 2)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f2[p][q])
x[i][v][f] <= 7;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f ==2 &&
3<=v<=4 && p== 2)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f2[p][q])
x[i][v][f] <= 6;


     forall( v in vcounter, f in fcounter, p in pcounter8 : f ==3 &&
3<=v<=4 && p!= 1 && p!=3)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f3[p][q])
x[i][v][f] <= 7;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f ==3 &&
3<=v<=4 && p== 1 && p==3)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f3[p][q])
x[i][v][f] <= 6;


    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==4 &&
3<=v<=4 && p!= 2)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f4[p][q])
x[i][v][f] <= 7;

    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==4 &&
3<=v<=4 && p== 2)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f4[p][q])
x[i][v][f] <= 6;


    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==5 &&
3<=v<=4 && p!= 2)
```

```
        sum(i in ccounter, q in qcounter5 : i == RndCov14_f5[p][q])
x[i][v][f] <= 7;

    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==5 &&
3<=v<=4 && p== 2)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f5[p][q])
x[i][v][f] <= 6;




    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==6 &&
3<=v<=4 && p!= 4)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f6[p][q])
x[i][v][f] <= 7;

    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==6 &&
3<=v<=4 && p== 4)

        sum(i in ccounter, q in qcounter5 : i == RndCov14_f6[p][q])
x[i][v][f] <= 6;




   forall( v in vcounter, f in fcounter, p in pcounter2 : f ==7 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter1 : i == RndCov14_f7[p][q])
x[i][v][f] <= 3;


   forall( v in vcounter, f in fcounter, p in pcounter2 : f ==8 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter1 : i == RndCov14_f8[p][q])
x[i][v][f] <= 3;


   forall( v in vcounter, f in fcounter, p in pcounter2 : f ==9 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter1 : i == RndCov14_f9[p][q])
x[i][v][f] <= 3;


   forall( v in vcounter, f in fcounter, p in pcounter2 : f ==10 &&
3<=v<=4)
```

```
        sum(i in ccounter,q in qcounter1 : i == RndCov14_f10[p][q])
x[i][v][f] <= 3;


    forall( v in vcounter, f in fcounter, p in pcounter2 : f ==11 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter1 : i == RndCov14_f11[p][q])
x[i][v][f] <= 3;


      forall( v in vcounter, f in fcounter, p in pcounter4 : f ==12 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter2 : i == RndCov14_f12[p][q])
x[i][v][f] <= 2;


   forall( v in vcounter, f in fcounter, p in pcounter4 : f ==13 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter2 : i == RndCov14_f13[p][q])
x[i][v][f] <= 2;


   forall( v in vcounter, f in fcounter, p in pcounter7 : f ==14 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter2 : i == RndCov14_f14[p][q])
x[i][v][f] <= 2;



        //cap20


   forall( v in vcounter, f in fcounter, p in pcounter9 : f ==2 &&
1<=v<=2)

        sum(i in ccounter,q in qcounter6 : i == RndCov20_f2[p][q])
x[i][v][f] <= 9;


   forall( v in vcounter, f in fcounter, p in pcounter9 : f ==3 &&
1<=v<=2)

        sum(i in ccounter,q in qcounter7 : i == RndCov20_f3[p][q])
x[i][v][f] <= 10;
```

```
   forall( v in vcounter, f in fcounter, p in pcounter9 : f ==4 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter7 : i == RndCov20_f4[p][q])
x[i][v][f] <= 10;


   forall( v in vcounter, f in fcounter, p in pcounter9 : f ==5 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter7 : i == RndCov20_f5[p][q])
x[i][v][f] <= 10;


  forall( v in vcounter, f in fcounter, p in pcounter9 : f ==6 && 1<=v<=2)

         sum(i in ccounter,q in qcounter7 : i == RndCov20_f6[p][q])
x[i][v][f] <= 10;


  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==7 && 1<=v<=2
&& p!=3 && p!=4)

         sum(i in ccounter,q in qcounter4 : i == RndCov20_f7[p][q])
x[i][v][f] <= 5;

  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==7 && 1<=v<=2
&& p==3 && p==4)

         sum(i in ccounter,q in qcounter4 : i == RndCov20_f7[p][q])
x[i][v][f] <= 4;


  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==8 &&
1<=v<=2&& p!=1 && p!=5)

         sum(i in ccounter,q in qcounter4 : i == RndCov20_f8[p][q])
x[i][v][f] <= 5;

 forall( v in vcounter, f in fcounter, p in pcounter7 : f ==8 && 1<=v<=2&&
p==1 && p==5)

         sum(i in ccounter,q in qcounter4 : i == RndCov20_f8[p][q])
x[i][v][f] <= 4;


  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==9 && 1<=v<=2
&& p!=4 && p!=5 && p!=6)
```

```
        sum(i in ccounter,q in qcounter4 : i == RndCov20_f9[p][q])
x[i][v][f] <= 5;

  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==9 && 1<=v<=2
&& p==4 && p==5 && p==6)

        sum(i in ccounter,q in qcounter4 : i == RndCov20_f9[p][q])
x[i][v][f] <= 4;



  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==10 &&
1<=v<=2&& p!=3 && p!=5)

        sum(i in ccounter,q in qcounter4 : i == RndCov20_f10[p][q])
x[i][v][f] <= 5;

  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==10 &&
1<=v<=2&& p==3 && p==5)

        sum(i in ccounter,q in qcounter4 : i == RndCov20_f10[p][q])
x[i][v][f] <= 4;



  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==11 && 1<=v<=2
&& p!=3 && p!=5)

        sum(i in ccounter,q in qcounter4 : i == RndCov20_f11[p][q])
x[i][v][f] <= 5;

  forall( v in vcounter, f in fcounter, p in pcounter7 : f ==11 &&
1<=v<=2&& p==3 && p==5)

        sum(i in ccounter,q in qcounter4 : i == RndCov20_f11[p][q])
x[i][v][f] <= 4;



  forall( v in vcounter, f in fcounter, p in pcounter2 : f ==12 && 1<=v<=2
&& p!=9)

        sum(i in ccounter,q in qcounter1 : i == RndCov20_f12[p][q])
x[i][v][f] <= 3;

  forall( v in vcounter, f in fcounter, p in pcounter2 : f ==12 && 1<=v<=2
&& p==9)

        sum(i in ccounter,q in qcounter1 : i == RndCov20_f12[p][q])
x[i][v][f] <= 2;



  forall( v in vcounter, f in fcounter, p in pcounter2 : f ==13 &&
1<=v<=2)
```

```
        sum(i in ccounter,q in qcounter1 : i == RndCov20_f13[p][q])
x[i][v][f] <= 3;


  forall( v in vcounter, f in fcounter, p in pcounter2 : f ==14 &&
1<=v<=2)

        sum(i in ccounter,q in qcounter1 : i == RndCov20_f14[p][q])
x[i][v][f] <= 3;


  forall( v in vcounter, f in fcounter, p in pcounter2 : f ==15 &&
1<=v<=2)

        sum(i in ccounter,q in qcounter1 : i == RndCov20_f15[p][q])
x[i][v][f] <= 3;


  forall( v in vcounter, f in fcounter, p in pcounter2 : f ==16 &&
1<=v<=2)

        sum(i in ccounter,q in qcounter1 : i == RndCov20_f16[p][q])
x[i][v][f] <= 3;


  forall( v in vcounter, f in fcounter, p in pcounter6 : f ==17 &&
1<=v<=2)

        sum(i in ccounter,q in qcounter2 : i == RndCov20_f17[p][q])
x[i][v][f] <= 2;




   //cap21


    forall( v in vcounter, f in fcounter, p in pcounter9 : 2<=f<=6 &&
3<=v<=4){

        sum(i in ccounter,q in qcounter7 : i == RndCov21_f2[p][q])
x[i][v][2] <= q7-1;

        sum(i in ccounter,q in qcounter7 : i == RndCov21_f3[p][q])
x[i][v][3] <= q7-1;

        sum(i in ccounter,q in qcounter7 : i == RndCov21_f4[p][q])
x[i][v][4] <= q7-1;

        sum(i in ccounter,q in qcounter7 : i == RndCov21_f5[p][q])
x[i][v][5] <= q7-1;
```

```
        sum(i in ccounter,q in qcounter7 : i == RndCov21_f6[p][q])
x[i][v][6] <= q7-1;

      }


    forall( v in vcounter, f in fcounter, p in pcounter7 : 7<=f<=11 &&
3<=v<=4){

        sum(i in ccounter,q in qcounter4 : i == RndCov21_f7[p][q])
x[i][v][7] <= q4-1;

        sum(i in ccounter,q in qcounter4 : i == RndCov21_f8[p][q])
x[i][v][8] <= q4-1;

        sum(i in ccounter,q in qcounter4 : i == RndCov21_f11[p][q])
x[i][v][11] <= q4-1;

      }


    forall( v in vcounter, f in fcounter, p in pcounter7 : f == 9 &&
3<=v<=4 && p!=4)

       sum(i in ccounter,q in qcounter4 : i == RndCov21_f9[p][q])
x[i][v][f] <= q4-1;

    forall( v in vcounter, f in fcounter, p in pcounter7 : f == 9 &&
3<=v<=4 && p==4)

       sum(i in ccounter,q in qcounter4 : i == RndCov21_f9[p][q])
x[i][v][f] <= q4-2;


    forall( v in vcounter, f in fcounter, p in pcounter7 : f == 10 &&
3<=v<=4 && p!=5)

       sum(i in ccounter,q in qcounter4 : i == RndCov21_f10[p][q])
x[i][v][f] <= q4-1;

    forall( v in vcounter, f in fcounter, p in pcounter7 : f == 10 &&
3<=v<=4 && p==5)

       sum(i in ccounter,q in qcounter4 : i == RndCov21_f10[p][q])
x[i][v][f] <= q4-2;



    forall( v in vcounter, f in fcounter, p in pcounter2 : 12<=f<=16 &&
3<=v<=4 && f !=15){
```

```
        sum(i in ccounter,q in qcounter1 : i == RndCov21_f12[p][q])
x[i][v][12] <= q1-1;

        sum(i in ccounter,q in qcounter1 : i == RndCov21_f13[p][q])
x[i][v][13] <= q1-1;

        sum(i in ccounter,q in qcounter1 : i == RndCov21_f14[p][q])
x[i][v][14] <= q1-1;

        sum(i in ccounter,q in qcounter1 : i == RndCov21_f16[p][q])
x[i][v][16] <= q1-1;

      }

    forall( v in vcounter, f in fcounter, p in pcounter1 : f == 15 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter1 : i == RndCov21_f15[p][q])
x[i][v][15] <= q1-1;


    forall( v in vcounter, f in fcounter, p in pcounter10 : f ==17 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter2 : i == RndCov21_f17[p][q])
x[i][v][f] <= q2-1;



    //cap27


     forall( v in vcounter, f in fcounter, p in pcounter9 : 2<=f<=6 &&
1<=v<=2){

        sum(i in ccounter,q in qcounter8 : i == RndCov27_f2[p][q])
x[i][v][2] <= q8-1;

        sum(i in ccounter,q in qcounter8 : i == RndCov27_f3[p][q])
x[i][v][3] <= q8-1;

        sum(i in ccounter,q in qcounter8 : i == RndCov27_f4[p][q])
x[i][v][4] <= q8-1;

        sum(i in ccounter,q in qcounter8 : i == RndCov27_f5[p][q])
x[i][v][5] <= q8-1;

        sum(i in ccounter,q in qcounter8 : i == RndCov27_f6[p][q])
x[i][v][6] <= q8-1;

      }
```

```
    /* Random covers for 50D*/


    //cap7

      forall( v in vcounter, f in fcounter, p in pcounter9 : f ==1 &&
3<=v<=4)

         sum(i in ccounter,q in qcounter11 : i == RndCov7_f1_50[p][q])
x[i][v][f] <= q11-1;



      forall( v in vcounter, f in fcounter, p in pcounter10 : 2<=f<=6 &&
3<=v<=4 && f !=3 && f !=5){

         sum(i in ccounter,q in qcounter2 : i == RndCov7_f2_50[p][q])
x[i][v][2] <= q2-1;

         sum(i in ccounter,q in qcounter2 : i == RndCov7_f4_50[p][q])
x[i][v][4] <= q2-1;

         sum(i in ccounter,q in qcounter2 : i == RndCov7_f6_50[p][q])
x[i][v][6] <= q2-1;

       }



     forall( v in vcounter, f in fcounter, p in pcounter4 :  3<=v<=4 && f
== 3 && f == 5){

        sum(i in ccounter,q in qcounter2 : i == RndCov7_f3_50[p][q])
x[i][v][3] <= q2-1;

        sum(i in ccounter,q in qcounter2 : i == RndCov7_f5_50[p][q])
x[i][v][5] <= q2-1;

      }



     forall( v in vcounter, f in fcounter, p in pcounter14 : f ==7 &&
3<=v<=4 && 1<=p<=6)

         sum(i in ccounter,q in qcounter2 : i == RndCov7_f7_50[p][q])
x[i][v][f] <= q2-1;

     forall( v in vcounter, f in fcounter, p in pcounter14 : f ==7 &&
3<=v<=4 && p>=7)

         sum(i in ccounter,q in qcounter2 : i == RndCov7_f7_50[p][q])
x[i][v][f] <= q3-1;
```

```
     forall( v in vcounter, f in fcounter, p in pcounter15 : f ==8 &&
3<=v<=4)

         sum(i in ccounter,q in qcounter3 : i == RndCov7_f8_50[p][q])
x[i][v][f] <= q3-1;



    forall( v in vcounter, f in fcounter, p in pcounter8 : f ==9 &&
3<=v<=4)

         sum(i in ccounter,q in qcounter3 : i == RndCov7_f9_50[p][q])
x[i][v][f] <= q3-1;



     //cap10

    forall( v in vcounter, f in fcounter, p in pcounter2 : 2<=f<=6 &&
1<=v<=2 && f != 5){

         sum(i in ccounter,q in qcounter1 : i == RndCov10_f2_50[p][q])
x[i][v][2] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov10_f3_50[p][q])
x[i][v][3] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov10_f4_50[p][q])
x[i][v][4] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov10_f6_50[p][q])
x[i][v][6] <= q1-1;

    }

     forall( v in vcounter, f in fcounter, p in pcounter1 : 1<=v<=2 && f
== 5)

         sum(i in ccounter,q in qcounter1 : i == RndCov10_f5_50[p][q])
x[i][v][5] <= q1-1;



     forall( v in vcounter, f in fcounter, p in pcounter16 : f ==7 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter3 : i == RndCov10_f7_50[p][q])
x[i][v][f] <= q3-1;



     forall( v in vcounter, f in fcounter, p in pcounter17 : f ==8 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter3 : i == RndCov10_f8_50[p][q])
x[i][v][f] <= q3-1;
```

```
     forall( v in vcounter, f in fcounter, p in pcounter18 : f ==9 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter3 : i == RndCov10_f9_50[p][q])
x[i][v][f] <= q3-1;



     forall( v in vcounter, f in fcounter, p in pcounter9 : f ==10 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter3 : i == RndCov10_f10_50[p][q])
x[i][v][f] <= q3-1;



     //cap14



     forall( v in vcounter, f in fcounter, p in pcounter11 : 2<=f<=6 && f
!=4 && f!=6 && 3<=v<=4){

         sum(i in ccounter,q in qcounter10 : i == RndCov14_f2_50[p][q])
x[i][v][2] <= q10-1;

         sum(i in ccounter,q in qcounter10 : i == RndCov14_f3_50[p][q])
x[i][v][3] <= q10-1;

         sum(i in ccounter,q in qcounter10 : i == RndCov14_f5_50[p][q])
x[i][v][5] <= q10-1;

       }



      forall( v in vcounter, f in fcounter, p in pcounter6 :  f ==4 &&
f==6 && 3<=v<=4){

         sum(i in ccounter,q in qcounter4 : i == RndCov14_f4_50[p][q])
x[i][v][4] <= q4-1;

         sum(i in ccounter,q in qcounter4 : i == RndCov14_f6_50[p][q])
x[i][v][6] <= q4-1;

             }



    forall( v in vcounter, f in fcounter, p in pcounter4 :  7<=f<=11 &&
3<=v<=4 && f != 11){

         sum(i in ccounter,q in qcounter2 : i == RndCov14_f7_50[p][q])
x[i][v][7] <= q2-1;
```

```
        sum(i in ccounter,q in qcounter2 : i == RndCov14_f8_50[p][q])
x[i][v][8] <= q2-1;

        sum(i in ccounter,q in qcounter2 : i == RndCov14_f9_50[p][q])
x[i][v][9] <= q2-1;

        sum(i in ccounter,q in qcounter2 : i == RndCov14_f10_50[p][q])
x[i][v][10] <= q2-1;

        }


    forall( v in vcounter, f in fcounter, p in pcounter10 :  3<=v<=4 && f
== 11)

       sum(i in ccounter,q in qcounter2 : i == RndCov14_f11_50[p][q])
x[i][v][11] <= q2-1;


    forall( v in vcounter, f in fcounter, p in pcounter19 : f==12 &&
3<=v<=4)

        sum(i in ccounter,q in qcounter3 : i == RndCov14_f12_50[p][q])
x[i][v][7] <= q3-1;



    //cap20

    forall( v in vcounter, f in fcounter, p in pcounter8 :  2<=f<=3 &&
1<=v<=2){

        sum(i in ccounter,q in qcounter9 : i == RndCov20_f2_50[p][q])
x[i][v][2] <= q9-1;

        sum(i in ccounter,q in qcounter9 : i == RndCov20_f3_50[p][q])
x[i][v][3] <= q9-1;

      }

     forall( v in vcounter, f in fcounter, p in pcounter8 :  4<=f<=6 &&
1<=v<=2){

        sum(i in ccounter,q in qcounter5 : i == RndCov20_f4_50[p][q])
x[i][v][4] <= q5-1;

        sum(i in ccounter,q in qcounter5 : i == RndCov20_f5_50[p][q])
x[i][v][5] <= q5-1;

        sum(i in ccounter,q in qcounter5 : i == RndCov20_f6_50[p][q])
x[i][v][6] <= q5-1;

        }
```

```
    forall( v in vcounter, f in fcounter, p in pcounter2 :  7<=f<=11 &&
1<=v<=2){

         sum(i in ccounter,q in qcounter1 : i == RndCov20_f7_50[p][q])
x[i][v][7] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov20_f8_50[p][q])
x[i][v][8] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov20_f9_50[p][q])
x[i][v][9] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov20_f10_50[p][q])
x[i][v][10] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov20_f11_50[p][q])
x[i][v][11] <= q1-1;

      }

      forall( v in vcounter, f in fcounter, p in pcounter4 :  12<=f<=13 &&
1<=v<=2){

         sum(i in ccounter,q in qcounter2 : i == RndCov20_f12_50[p][q])
x[i][v][12] <= q2-1;

         sum(i in ccounter,q in qcounter2 : i == RndCov20_f13_50[p][q])
x[i][v][13] <= q2-1;

      }

      forall( v in vcounter, f in fcounter, p in pcounter20 :  f == 14 &&
1<=v<=2)

         sum(i in ccounter,q in qcounter2 : i == RndCov20_f14_50[p][q])
x[i][v][14] <= q2-1;




    //cap21

     forall( v in vcounter, f in fcounter, p in pcounter8 : f==2 &&
3<=v<=4 && p!=4)

         sum(i in ccounter,q in qcounter5 : i == RndCov21_f2_50[p][q])
x[i][v][2] <= q5-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f==2 &&
3<=v<=4 && p==4)

         sum(i in ccounter,q in qcounter5 : i == RndCov21_f2_50[p][q])
x[i][v][2] <= q9-1;
```

```
     forall( v in vcounter, f in fcounter, p in pcounter8 :  f == 3 &&
3<=v<=4 && p!=2 && p!=3)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f3_50[p][q])
x[i][v][3] <= q5-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 3 &&
3<=v<=4 && p==2 && p==3)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f3_50[p][q])
x[i][v][3] <= q9-1;



     forall( v in vcounter, f in fcounter, p in pcounter8 :  f == 4 && p
!= 3)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f4_50[p][q])
x[i][v][4] <= q5-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 :  f == 4 && p
== 3)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f4_50[p][q])
x[i][v][4] <= q9-1;



     forall( v in vcounter, f in fcounter, p in pcounter8 :  f == 5 &&
3<=v<=4 && p!=2 && p!=4)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f5_50[p][q])
x[i][v][5] <= q5-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 :  f == 5 &&
3<=v<=4 && p==2 && p==4)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f5_50[p][q])
x[i][v][5] <= q9-1;



     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 6 &&
3<=v<=4 && p!=3)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f6_50[p][q])
x[i][v][6] <= q5-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 5 &&
3<=v<=4 && p ==3)

          sum(i in ccounter,q in qcounter5 : i == RndCov21_f6_50[p][q])
x[i][v][6] <= q9-1;
```

```
     forall( v in vcounter, f in fcounter, p in pcounter2 :  8<=f<=11 &&
3<=v<=4){

         sum(i in ccounter,q in qcounter1 : i == RndCov21_f8_50[p][q])
x[i][v][8] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov21_f9_50[p][q])
x[i][v][9] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov21_f10_50[p][q])
x[i][v][10] <= q1-1;

         sum(i in ccounter,q in qcounter1 : i == RndCov21_f11_50[p][q])
x[i][v][11] <= q1-1;

       }

       forall( v in vcounter, f in fcounter, p in pcounter1 : f == 7 &&
3<=v<=4)

         sum(i in ccounter,q in qcounter1 : i == RndCov21_f7_50[p][q])
x[i][v][7] <= q1-1;



      forall( v in vcounter, f in fcounter, p in pcounter4 :  12<=f<=13 &&
3<=v<=4){

         sum(i in ccounter,q in qcounter2 : i == RndCov21_f12_50[p][q])
x[i][v][12] <= q2-1;

         sum(i in ccounter,q in qcounter2 : i == RndCov21_f13_50[p][q])
x[i][v][13] <= q2-1;

       }

      forall( v in vcounter, f in fcounter, p in pcounter6 : f == 14 &&
3<=v<=4)

         sum(i in ccounter,q in qcounter2 : i == RndCov21_f14_50[p][q])
x[i][v][12] <= q2-1;



      //cap27

      forall( v in vcounter, f in fcounter, p in pcounter8 : f == 2 &&
1<=v<=2 && p!=2 && p!=3 )

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f2_50[p][q])
x[i][v][f] <= q6-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 2 &&
1<=v<=2 && p==2 && p==3)

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f2_50[p][q])
x[i][v][f] <= q12-1;
```

```
     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 3 &&
1<=v<=2 && p==3 )

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f3_50[p][q])
x[i][v][f] <= q6-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 3 &&
1<=v<=2 && p!=3)

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f3_50[p][q])
x[i][v][f] <= q12-1;



     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 4 &&
1<=v<=2 && p==4 )

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f4_50[p][q])
x[i][v][f] <= q12-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 4 &&
1<=v<=2 && p!=4)

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f4_50[p][q])
x[i][v][f] <= q6-1;



     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 5 &&
1<=v<=2 && p!=1 && p!=3 )

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f5_50[p][q])
x[i][v][f] <= q6-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 5 &&
1<=v<=2 && p==1 && p==3)

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f5_50[p][q])
x[i][v][f] <= q12-1;



     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 6 &&
1<=v<=2 && p!=1 && p!=4 )

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f6_50[p][q])
x[i][v][f] <= q6-1;

     forall( v in vcounter, f in fcounter, p in pcounter8 : f == 6 &&
1<=v<=2 && p==1 && p==4)

         sum(i in ccounter,q in qcounter6 : i == RndCov27_f6_50[p][q])
x[i][v][f] <= q12-1;
```

```
     forall( v in vcounter, f in fcounter, p in pcounter11 : f == 7 &&
1<=v<=2 && p!=5 && p!=6 )

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f7_50[p][q])
x[i][v][f] <= q10 -1 ;

     forall( v in vcounter, f in fcounter, p in pcounter11 : f == 7 &&
1<=v<=2 && p==5)

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f7_50[p][q])
x[i][v][f] <= q3 -1;

    forall( v in vcounter, f in fcounter, p in pcounter11 : f == 7 &&
1<=v<=2 && p==6)

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f7_50[p][q])
x[i][v][f] <= q2 -1;



    forall( v in vcounter, f in fcounter, p in pcounter11 :  8<=f<=11 &&
1<=v<=2){

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f8_50[p][q])
x[i][v][8] <= q10-1;

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f9_50[p][q])
x[i][v][9] <= q10-1;

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f10_50[p][q])
x[i][v][10] <= q10-1;

         sum(i in ccounter,q in qcounter10 : i == RndCov27_f11_50[p][q])
x[i][v][11] <= q10-1;

       }



     forall( v in vcounter, f in fcounter, p in pcounter2 : f == 12 &&
1<=v<=2 && p!=1 && p!=3 && p!=7 && p!=8 && p!=9)

         sum(i in ccounter,q in qcounter1 : i == RndCov27_f12_50[p][q])
x[i][v][f] <= q1-1;

     forall( v in vcounter, f in fcounter, p in pcounter2 : f == 12 &&
1<=v<=2 && p ==1 && p ==3 && p ==7 && p ==8 && p ==9)

         sum(i in ccounter,q in qcounter1 : i == RndCov27_f12_50[p][q])
x[i][v][f] <= q2-1;



    forall( v in vcounter, f in fcounter, p in pcounter2 : f == 13 &&
1<=v<=2 && p!=3 && p!=4 && p!=5 && p!=6 && p!=10 && p!=11)
```

```
        sum(i in ccounter,q in qcounter1 : i == RndCov27_f13_50[p][q])
x[i][v][f] <= q1-1;

    forall( v in vcounter, f in fcounter, p in pcounter2 : f == 13 &&
1<=v<=2 && p ==3 && p ==4 && p ==5 && p ==6 && p ==10 && p==11 )

        sum(i in ccounter,q in qcounter1 : i == RndCov27_f13_50[p][q])
x[i][v][f] <= q2-1;



       forall( v in vcounter, f in fcounter, p in pcounter10 : f == 14 &&
1<=v<=2 && p!=1 && p!=5 && p!=6 && p!=7 && p!=8 && p!=10&& p!=11)

        sum(i in ccounter,q in qcounter1 : i == RndCov27_f14_50[p][q])
x[i][v][f] <= q1-1;

    forall( v in vcounter, f in fcounter, p in pcounter10 : f == 14 &&
1<=v<=2 && p ==1 && p ==5 && p ==6 && p ==7 && p ==8 && p==10 && p==11)

        sum(i in ccounter,q in qcounter1 : i == RndCov27_f14_50[p][q])
x[i][v][f] <= q2-1;



    forall( v in vcounter, f in fcounter, p in pcounter1 : f == 15 &&
1<=v<=2 && p!=1 && p!=2 && p!=5 && p!=7)

        sum(i in ccounter,q in qcounter1 : i == RndCov27_f15_50[p][q])
x[i][v][f] <= q1-1;

    forall( v in vcounter, f in fcounter, p in pcounter1 : f == 15 &&
1<=v<=2 && p ==1 && p ==2 && p ==5 && p ==7)

        sum(i in ccounter,q in qcounter1 : i == RndCov27_f15_50[p][q])
x[i][v][f] <= q2-1;



      }
```

Appendix2

JAVA CODE FOR FIX & OPTIMIZE HEURISTIC

```
import ilog.concert.*;
import ilog.cplex.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class F_And_O_Second_Step_Test1 {

public static int n = 20;// number of customers
public static   int v = 4; // number of vehicles
public static   int F = 21;// number of available frequencies
public static   int D = 5; // number of days of the week
public static   int H =52;// number of weeks in a year
public static    int h1 = 300;
//public static    int h1 = 600;
```

```
public static    int smax = 5;// maximum number of customers being visited
on a route

public static   int m = 21; // big m

public static   int F1[] = {0,1,6,11,16};

public static   int F2[] = {0,2,7,12,17};

public static   int F3[] = {0,3,8,13,18};

public static   int F4[] = {0,4,9,14,19};

public static   int F5[] = {0,5,10,15,20};

public static   double r[] = {8.4,8.4,7,7};

//public static    double r[] = {9.8,9.8,7,7};

public static   int g[] = {17,17,17,17};

public static   int a[] = {48960,48960,40800,40800};

//public static int a[] = {57120,57120,40800,40800};

public static   int c[]= {10,10,7,7};

//public static int c[]= {20,20,14,14};

//public static int c[]= {27,27,21,21};

public          static                              int          p[]          =
{365,52,52,52,52,52,26,26,26,26,26,18,18,18,18,18,13,13,13,13,13};

public static   int delta = 5;

public static int step = 0;

public static int binarizer = 0;

public static int improver = 0;

public static double newobjective = 0;

public static double newobjectivep = 0;

public static double fixedXValues[][][] = new double[n][v][F];

public static double fixedLValues[][] = new double[v][F];

public static double fixedRValues[][][] = new double[D][v][H];

public static double fixedCValues[][] = new double[v][F];

public static double fixedVValues[] = new double[v];
```

```
public static IloNumVar [][] L = new IloNumVar[v][F];
public static IloNumVar [][] BL = new IloNumVar[v][F];
public static IloNumVar [][] DL = new IloNumVar[v][F];

public static IloNumVar [][] C = new IloNumVar[v][F];

public static IloNumVar [] V = new IloNumVar [v];
public static IloNumVar [] BV = new IloNumVar [v];
public static IloNumVar [] DV = new IloNumVar [v];

public static IloNumVar [][][] R = new IloNumVar[D][v][H];
public static IloNumVar [][][] BR = new IloNumVar[D][v][H];
public static IloNumVar [][][] DR = new IloNumVar[D][v][H];

public static IloNumVar [][][] x = new IloNumVar[n][v][F];
public static IloNumVar [][][] Bx = new IloNumVar[n][v][F];
public static IloNumVar [][][] Dx = new IloNumVar[n][v][F];


public static void main(String[] args)throws Exception {

    int i=0;
    int f =0;
    File excel = new File("C:\\Book1.xlsx");
    FileInputStream fis = new FileInputStream(excel);
    XSSFWorkbook wb = new XSSFWorkbook(fis);
    XSSFSheet ws = wb.getSheet("Sheet6");
```

```
    int rowNum = ws.getLastRowNum() + 1;
    int colNum = ws.getRow(0).getLastCellNum();

    double y[][] = new double[rowNum][colNum];

    for( i=0;i<rowNum;i++){
        XSSFRow row = ws.getRow(i);
        for(f=0;f<colNum;f++){
            XSSFCell cell = row.getCell(f);
            double value = cellToDouble(cell);
            y[i][f]= value;
    //  System.out.print(y[i][f] + "\t\t" );
        }

    //  System.out.println(" " );
    }
//  System.out.println(y[2][4]);

    int k=0;
    int l =0;
    File excel1 = new File("C:\\Book1.xlsx");
    FileInputStream fis1 = new FileInputStream(excel1);
    XSSFWorkbook wb1 = new XSSFWorkbook(fis1);
    XSSFSheet ws1 = wb1.getSheet("Sheet2");

    int rowNum1 = ws1.getLastRowNum() + 1;
    int colNum1 = ws1.getRow(0).getLastCellNum();

    double K[][] = new double[rowNum1][colNum1];
```

```
for( k=0;k<rowNum1;k++){
    XSSFRow row1 = ws1.getRow(k);
    for(l=0;l<colNum1;l++){
        XSSFCell cell1 = row1.getCell(l);
        double value = cellToDouble(cell1);
        K[k][l]= value;
//  System.out.print(K[k][l] + "\t\t" );
    }

//  System.out.println(" " );
}
int s=0;
int q =0;
File excel2 = new File("C:\\Book4.xlsx");
FileInputStream fis2 = new FileInputStream(excel2);
XSSFWorkbook wb2 = new XSSFWorkbook(fis2);
XSSFSheet ws2 = wb2.getSheet("Sheet7");

int rowNum2 = ws2.getLastRowNum() + 1;
int colNum2 = ws2.getRow(0).getLastCellNum();

double b[][] = new double[rowNum2][colNum2];

for( s=0;s<rowNum2;s++){
    XSSFRow row = ws2.getRow(s);
    for(q=0;q<colNum2;q++){
        XSSFCell cell = row.getCell(q);
        double value = cellToDouble(cell);
```

```
            b[s][q]= value;
        //  System.out.print(b[s][q] + "\t\t"  );


        }
    //  System.out.println(" " );
    }

    Model(n,v,F,F1,F2,F3,F4,F5,D,H,r,g,a,y,h1,K,smax,c,m,p,b );
}

private static int cellToInteger(XSSFCell cell) {
    int type;
    int result1;
    type = cell.getCellType();
    switch(type){
    case 0 :
        result1 = (int) cell.getNumericCellValue();
        break;
    case 1:
        result1=Integer.parseInt(cell.getStringCellValue()) ;
        break;
        default:
            throw new RuntimeException(" There is no support for this type
of cell");

    }
    return result1;
}
```

```
private static double cellToDouble(XSSFCell cell) {

    int type;

    double result;

    type = cell.getCellType();

    switch(type){

    case 0 :

        result = (double) cell.getNumericCellValue();

        break;

    case 1:

        result =Double.parseDouble(cell.getStringCellValue()) ;

        break;

        default:

            throw new RuntimeException(" There is no support for this type
of cell");



    }


    return result  ;

}




public static void Model(int n,int v, int F,int F1[],int F2[],int F3[],
int F4[],int F5[],int D, int H, double r[], int g[], int a[], double
y[][], int h1, double[][] K, int smax, int c[], int m, int p[], double[][]
b ){

    try{


        //creating an empty model

        IloCplex model = new IloCplex();

        for(int i = 0; i<n; i++){

            for(int j = 0; j<v;j++){
```

```
        for( int f = 0; f<F; f++){
        x[i][j][f] =  model.numVar(0,1);
        }
    }
}
   for(int i = 0; i<n; i++){
    for(int j = 0; j<v;j++){
        for( int f = 0; f<F; f++){
        Bx[i][j][f] =  model.boolVar();
        }
    }
}
for(int i = 0; i<n; i++){
    for(int j = 0; j<v;j++){
        for( int f = 0; f<F; f++){
        Dx[i][j][f] =  model.numVar(0, 1);
        }
    }
}
// defining L[v][f] as Double
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        L[j][f] =model.numVar(0,1);
    }
}
//defining L[v][f]'s boolean part
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        BL[j][f] = model.boolVar();
```

```
    }
}
//defining L[v][f]'s double part
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        DL[j][f] = model.numVar(0,1);
    }
}
// defining C[v][f] as integer
for(int j = 0; j < v; j++){
    for(int f = 0 ; f< F; f++){
        C[j][f] = model.numVar(0,Double.MAX_VALUE);
    }
}
//defining V[v] as double
for(int j = 0 ; j< v ; j++){
    V[j] =  model.boolVar();
}
//defining R[d][v][H] as double
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            R[d][j][l] =model.numVar(0,1);
        }
    }
}
// defining R[d][v][H]'s boolean part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
```

```
        for(int l= 0; l< H; l++){
            BR[d][j][l] = model.boolVar();
        }
    }
}
//defining R[d][v][H]'s double part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            DR[d][j][l] =model.numVar(0,1);
        }
    }
}
//objective
IloLinearNumExpr obj = model.linearNumExpr();
// adding first objective element
for(int j = 0; j<v;j++){
    for(int f = 0; f<F;f++){
        if(f == 0){
    int t = g[j] * p[f];
    obj.addTerm(t,L[j][f]);
//  System.out.println(t);
    }else{
        obj.addTerm(0, L[j][f]);
    }
}
}
//adding second objective element
for(int d=0; d<D;d++){
```

```
    for(int l=0;l<H;l++){
        for(int j=0;j<v;j++){
            obj.addTerm(g[j], R[d][j][l]);
        }
    }
}
//adding third objective element
for(int j=0; j<v;j++){
    for(int f=0;f<F;f++){
        double S = r[j] * p[f];
        obj.addTerm(S, C[j][f]);
    }
}
//adding fourth objective element
for(int j= 0; j<v; j++){
    obj.addTerm(a[j], V[j]);
}
// adding holding cost to objective
for(int i =0; i<n;i++){
    for(int j = 0; j<v;j++){
        for(int f=0; f<F; f++){
            double HoldingCost = y[i][f] * h1*0.5;
            obj.addTerm(HoldingCost,x[i][j][f]);

        }
    }
}
//adding replenishment cost
for(int i=0;i<n;i++){
```

```
            for(int j=0; j<v;j++){
                for(int f=0; f<F; f++){
                    obj.addTerm(K[i][f], x[i][j][f]);
                }
            }
        }
//  System.out.println(obj);
        model.addMinimize(obj);

for(int w =0; w<5;w++){
    for(int fixer =0; fixer<n;fixer = fixer+delta){
        //resolving customers 0-4
        if(step==5){
            for(int i = 0; i<n; i++){
                for(int j = 0; j<v;j++){
                    for( int f = 0; f<F; f++){
                    Bx[i][j][f] =  model.boolVar();
                    }
                }
            }
            for(int i = 0; i<n; i++){
                for(int j = 0; j<v;j++){
                    for( int f = 0; f<F; f++){
                    Dx[i][j][f] =  model.numVar(0, 1);
                    }
                }
            }
            // defining L[v][f] as Double
            for(int j = 0; j <v; j++){
```

```
                for(int f = 0; f<F; f++){
                    L[j][f] =model.numVar(0,1);
                }
            }
            // defining C[v][f] as integer
            for(int j = 0; j < v; j++){
                for(int f = 0 ; f< F; f++){
                    C[j][f] = model.numVar(0,Double.MAX_VALUE);
                }
            }
            //defining V[v] as double
            for(int j = 0 ; j< v ; j++){
                V[j] =  model.numVar(0,1);
            }
            //defining R[d][v][H] as double
            for(int d = 0; d<D; d++){
                for(int j = 0 ; j <v; j++){
                    for(int l= 0; l< H; l++){
                        R[d][j][l] =model.numVar(0,1);
                    }
                }
            }
            for(int i = fixer;i<fixer+delta;i++){
                for(int j =0; j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                constraint312=
model.linearNumExpr();
                        constraint312.addTerm(1.0, Dx[i][j][f]);
                        model.addEq(constraint312,0 );
```

```
                        }
                    }


                }
                for(int i = fixer+delta;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }
            }
            //Binarizing the resolved problem for customers 0-4
            if(step == 6 && binarizer == 1){


                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
```

```
            for( int f = 0; f<F; f++){
            Dx[i][j][f] =  model.numVar(0, 1);
            }
        }
    }
    //defining L[v][f]'s boolean part
    for(int j = 0; j <v; j++){
        for(int f = 0; f<F; f++){
            BL[j][f] = model.boolVar();
        }
    }
    //defining L[v][f]'s double part
    for(int j = 0; j <v; j++){
        for(int f = 0; f<F; f++){
            DL[j][f] = model.numVar(0,1);
        }
    }
    //defining V[v]'s boolean part
        for(int j = 0 ; j< v ; j++){
            BV[j] = model.boolVar();
                }
    //defining V[v]'s double part
        for(int j = 0 ; j< v ; j++){
            DV[j] =  model.numVar(0,1);
                }

    for(int d = 0; d<D; d++){
        for(int j = 0 ; j <v; j++){
            for(int l= 0; l< H; l++){
```

```
                    BR[d][j][l] = model.boolVar();
                }
            }
        }
        //defining R[d][v][H]'s double part
        for(int d = 0; d<D; d++){
            for(int j = 0 ; j <v; j++){
                for(int l= 0; l< H; l++){
                    DR[d][j][l] =model.numVar(0,1);
                }
            }
        }
        for(int j=0;j<v;j++){
            for(int f=0;f<F;f++){
                IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                constraintL.addTerm(1.0, L[j][f]);

model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
            }
        }
        for(int j=0;j<v;j++){
            for(int f=0;f<F;f++){
                IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                constraintL.addTerm(1.0, DL[j][f]);
                model.addEq(constraintL,0);
            }
        }
        for(int d = 0; d<D; d++){
```

```
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, R[d][j][l]);

model.addEq(constraintR,model.sum(BR[d][j][l],DR[d][j][l]) );
                        }
                    }
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, DR[d][j][l]);
                            model.addEq(constraintR,0 );
                        }
                    }
                }
                for(int i = 0;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
```

```
        }
    }
    // resolving problem for customers 5-9
/*  System.out.println("step: " + step);
    System.out.println("fixer: " + fixer);*/
    if(step == 7){

        for(int i = 0; i<n; i++){
            for(int j = 0; j<v;j++){
                for( int f = 0; f<F; f++){
                Bx[i][j][f] =  model.boolVar();
                }
            }
        }
        for(int i = 0; i<n; i++){
            for(int j = 0; j<v;j++){
                for( int f = 0; f<F; f++){
                Dx[i][j][f] =  model.numVar(0, 1);
                }
            }
        }
        // defining L[v][f] as Double
        for(int j = 0; j <v; j++){
            for(int f = 0; f<F; f++){
                L[j][f] =model.numVar(0,1);
            }
        }
        // defining C[v][f] as integer
        for(int j = 0; j < v; j++){
```

```
                for(int f = 0 ; f< F; f++){
                    C[j][f] = model.numVar(0,Double.MAX_VALUE);
                }
            }
            //defining V[v] as double
            for(int j = 0 ; j< v ; j++){
                V[j] =  model.numVar(0,Double.MAX_VALUE);
            }
            //defining R[d][v][H] as double
            for(int d = 0; d<D; d++){
                for(int j = 0 ; j <v; j++){
                    for(int l= 0; l< H; l++){
                        R[d][j][l] =model.numVar(0,Double.MAX_VALUE);
                    }
                }
            }
            for(int i = fixer-delta;i<fixer;i++){
                for(int j =0; j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                constraint312=
model.linearNumExpr();
                        constraint312.addTerm(1.0, Dx[i][j][f]);
                        model.addEq(constraint312,0 );
                    }
                }

            }
            for(int i = fixer;i<n;i++){
                for(int j =0; j<v;j++){
```

```
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                            //System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }
                for(int i = 0;i<fixer-delta;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        }
                    }
                }


            }
            //binarizing customers 5-9
            if(step == 8 && binarizer == 2){


                for(int i = 0;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
```

```
                        constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                    }
                }
            }
             for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Dx[i][j][f]      =              model.numVar(0,
Double.MAX_VALUE);
                        }
                    }
                }
                for(int i = 0;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr               constraintw=
model.linearNumExpr();
                            constraintw.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraintw,0 );
                        }
                    }
```

```
    }

//defining L[v][f]'s boolean part
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        BL[j][f] = model.boolVar();
    }
}
//defining L[v][f]'s double part
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        DL[j][f] = model.numVar(0,1);
    }
}
//defining V[v]'s boolean part
        for(int j = 0 ; j< v ; j++){
            BV[j] = model.boolVar();
        }
        //defining V[v]'s double part
        for(int j = 0 ; j< v ; j++){
            DV[j] =  model.numVar(0,1);
        }

// defining R[d][v][H]'s boolean part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            BR[d][j][l] = model.boolVar();
        }
```

```
                    }
                }
                //defining R[d][v][H]'s double part
                for(int d = 0; d<D; d++){
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            DR[d][j][l] =model.numVar(0,1);
                        }
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, L[j][f]);

model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, DL[j][f]);
                        model.addEq(constraintL,0);
                    }
                }
                for(int j=0;j<v;j++){
                        IloLinearNumExpr                    constraintV=
model.linearNumExpr();
                        constraintV.addTerm(1.0, V[j]);
```

```
                    model.addEq(constraintV,model.sum(BV[j],DV[j]));
                }
            for(int j=0;j<v;j++){
                IloLinearNumExpr constraintV= model.linearNumExpr();
                constraintV.addTerm(1.0, DV[j]);
                model.addEq(constraintV,0);
            }
            for(int d = 0; d<D; d++){
                for(int j=0;j<v;j++){
                    for(int l=0;l<H;l++){
                        IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                        constraintR.addTerm(1.0, R[d][j][l]);

model.addEq(constraintR,model.sum(BR[d][j][l],DR[d][j][l]) );
                    }
                }
            }
            for(int d = 0; d<D; d++){
                for(int j=0;j<v;j++){
                    for(int l=0;l<H;l++){
                        IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                        constraintR.addTerm(1.0, DR[d][j][l]);
                        model.addEq(constraintR,0 );
                    }
                }
            }

        }
```

```
//reoptimizing for customers 10-14
if(step == 9 && improver >0){

    for(int i = 0; i<n; i++){
        for(int j = 0; j<v;j++){
            for( int f = 0; f<F; f++){
            Bx[i][j][f] =  model.boolVar();
            }
        }
    }
    for(int i = 0; i<n; i++){
        for(int j = 0; j<v;j++){
            for( int f = 0; f<F; f++){
            Dx[i][j][f] =  model.numVar(0, 1);
            }
        }
    }
    // defining L[v][f] as Double
    for(int j = 0; j <v; j++){
        for(int f = 0; f<F; f++){
            L[j][f] =model.numVar(0,1);
        }
    }
    // defining C[v][f] as integer
    for(int j = 0; j < v; j++){
        for(int f = 0 ; f< F; f++){
            C[j][f] = model.numVar(0,Double.MAX_VALUE);
        }
    }
```

```
                //defining V[v] as double
                for(int j = 0 ; j< v ; j++){
                    V[j] =  model.numVar(0,Double.MAX_VALUE);
                }
                //defining R[d][v][H] as double
                for(int d = 0; d<D; d++){
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            R[d][j][l] =model.numVar(0,Double.MAX_VALUE);
                        }
                    }
                }
        /*      for(int i = 0;i<2*delta;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraint312,0 );
                        }
                    }

                }*/
                for(int i = 2*delta;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);
```

```
model.addEq(constraint312,fixedXValues[i][j][f] );
                            //System.out.println(fixedXValues[i][j][f]);
                        }
                    }
                }
            }
            //case without improvement customers 10-14
            if(step == 9 && improver == 0 ){

                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Dx[i][j][f] =  model.numVar(0, 1);
                        }
                    }
                }
                // defining L[v][f] as Double
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        L[j][f] =model.numVar(0,1);
                    }
```

```
                }
                // defining C[v][f] as integer
                for(int j = 0; j < v; j++){
                    for(int f = 0 ; f< F; f++){
                        C[j][f] = model.numVar(0,Double.MAX_VALUE);
                    }
                }
                //defining V[v] as double
                for(int j = 0 ; j< v ; j++){
                    V[j] =  model.numVar(0,Double.MAX_VALUE);
                }
                //defining R[d][v][H] as double
                for(int d = 0; d<D; d++){
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            R[d][j][l] =model.numVar(0,Double.MAX_VALUE);
                        }
                    }
                }
            /*  for(int i = 2*delta;i<3*delta;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                    constraintq=
model.linearNumExpr();
                            constraintq.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraintq,0 );
                        }
                    }
```

```
                }*/
                for(int i = 0;i<10;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                    constraintpr=
model.linearNumExpr();
                            constraintpr.addTerm(1.0, x[i][j][f]);
                            model.addEq(constraintpr,fixedXValues[i][j][f]
);
                        }
                    }
                }
                for(int i = 15;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                    constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }
            }


            //binarize 10-14
            if(step == 10 && binarizer == 3){
            /*  System.out.println("fixer:" + fixer);
                System.out.println("step:" + step);*/
```

```
                for(int i = 0;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }
            /*  for(int i = 0;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraint312,0 );
                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }*/
                 for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Bx[i][j][f] =  model.boolVar();
                            }
                        }
```

```
                    }
                    for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Dx[i][j][f]      =              model.numVar(0,
Double.MAX_VALUE);
                            }
                        }
                    }

                //defining L[v][f]'s boolean part
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        BL[j][f] = model.boolVar();
                    }
                }
                //defining L[v][f]'s double part
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        DL[j][f] = model.numVar(0,1);
                    }
                }
                //defining V[v]'s boolean part
                        for(int j = 0 ; j< v ; j++){
                            BV[j] = model.boolVar();
                        }
                        //defining V[v]'s double part
                        for(int j = 0 ; j< v ; j++){
                            DV[j] =  model.numVar(0,1);
```

```
                }


            // defining R[d][v][H]'s boolean part
            for(int d = 0; d<D; d++){
                for(int j = 0 ; j <v; j++){
                    for(int l= 0; l< H; l++){
                        BR[d][j][l] = model.boolVar();
                    }
                }
            }
            //defining R[d][v][H]'s double part
            for(int d = 0; d<D; d++){
                for(int j = 0 ; j <v; j++){
                    for(int l= 0; l< H; l++){
                        DR[d][j][l] =model.numVar(0,1);
                    }
                }
            }
            for(int j=0;j<v;j++){
                for(int f=0;f<F;f++){
                    IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                    constraintL.addTerm(1.0, L[j][f]);

model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
                }
            }
            for(int j=0;j<v;j++){
                for(int f=0;f<F;f++){
```

```
                        IloLinearNumExpr                        constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, DL[j][f]);
                        model.addEq(constraintL,0);
                    }
                }
                for(int j=0;j<v;j++){
                        IloLinearNumExpr                        constraintV=
model.linearNumExpr();
                        constraintV.addTerm(1.0, V[j]);
                        model.addEq(constraintV,model.sum(BV[j],DV[j]));
                    }
                for(int j=0;j<v;j++){
                    IloLinearNumExpr constraintV= model.linearNumExpr();
                    constraintV.addTerm(1.0, DV[j]);
                    model.addEq(constraintV,0);
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, R[d][j][l]);

model.addEq(constraintR,model.sum(BR[d][j][l],DR[d][j][l]) );
                        }
                    }
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
```

```
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, DR[d][j][l]);
                            model.addEq(constraintR,0 );
                        }
                    }
                }
            }
            //reoptimizing customers 15-19
            if(step == 11 && improver >0){
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Dx[i][j][f] =  model.numVar(0, 1);
                        }
                    }
                }
                // defining L[v][f] as Double
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        L[j][f] =model.numVar(0,1);
                    }
```

```
                }
                // defining C[v][f] as integer
                for(int j = 0; j < v; j++){
                    for(int f = 0 ; f< F; f++){
                        C[j][f] = model.numVar(0,Double.MAX_VALUE);
                    }
                }
                //defining V[v] as double
                for(int j = 0 ; j< v ; j++){
                    V[j] =  model.numVar(0,Double.MAX_VALUE);
                }
                //defining R[d][v][H] as double
                for(int d = 0; d<D; d++){
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            R[d][j][l] =model.numVar(0,Double.MAX_VALUE);
                        }
                    }
                }
            /*  for(int i = 0;i<fixer+delta;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraint312,0 );
                        }
                    }
```

```
                }*/
                for(int i = fixer+delta;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        }
                    }
                }
            }
            //reoptimizing 15-19
            if(step == 11 && improver == 0){
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Dx[i][j][f] =  model.numVar(0, 1);
                        }
                    }
                }
                // defining L[v][f] as Double
```

```
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        L[j][f] =model.numVar(0,1);
                    }
                }
                // defining C[v][f] as integer
                for(int j = 0; j < v; j++){
                    for(int f = 0 ; f< F; f++){
                        C[j][f] = model.numVar(0,Double.MAX_VALUE);
                    }
                }
                //defining V[v] as double
                for(int j = 0 ; j< v ; j++){
                    V[j] =  model.numVar(0,Double.MAX_VALUE);
                }
                //defining R[d][v][H] as double
                for(int d = 0; d<D; d++){
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            R[d][j][l] =model.numVar(0,Double.MAX_VALUE);
                        }
                    }
                }
            /*  for(int i = fixer+delta;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, Dx[i][j][f]);
```

```
                        model.addEq(constraint312,0 );
                    }
                }

            }*/
            for(int i =0 ;i<fixer+delta;i++){
                for(int j =0; j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                constraint312=
model.linearNumExpr();
                        constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                    }
                }
            }
        }
        if(step == 12 && binarizer == 4){
            for(int i = 0;i<n;i++){
                for(int j =0; j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                constraint312=
model.linearNumExpr();
                        constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                    }
                }
            }
        /*  for(int i = 0;i<n;i++){
                for(int j =0; j<v;j++){
```

```
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraint312,0 );
                        }
                    }
                }*/
                 for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Bx[i][j][f] =  model.boolVar();
                            }
                        }
                    }
                    for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Dx[i][j][f]      =             model.numVar(0,
Double.MAX_VALUE);
                            }
                        }
                    }

                //defining L[v][f]'s boolean part
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        BL[j][f] = model.boolVar();
                    }
                }
```

```
//defining L[v][f]'s double part
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        DL[j][f] = model.numVar(0,1);
    }
}
//defining V[v]'s boolean part
        for(int j = 0 ; j< v ; j++){
            BV[j] = model.boolVar();
        }
        //defining V[v]'s double part
        for(int j = 0 ; j< v ; j++){
            DV[j] =  model.numVar(0,1);
        }

// defining R[d][v][H]'s boolean part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            BR[d][j][l] = model.boolVar();
        }
    }
}
//defining R[d][v][H]'s double part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            DR[d][j][l] =model.numVar(0,1);
        }
```

```
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                        constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, L[j][f]);

model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                        constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, DL[j][f]);
                        model.addEq(constraintL,0);
                    }
                }
                for(int j=0;j<v;j++){
                        IloLinearNumExpr                        constraintV=
model.linearNumExpr();
                        constraintV.addTerm(1.0, V[j]);
                        model.addEq(constraintV,model.sum(BV[j],DV[j]));
                    }
                for(int j=0;j<v;j++){
                    IloLinearNumExpr constraintV= model.linearNumExpr();
                    constraintV.addTerm(1.0, DV[j]);
                    model.addEq(constraintV,0);
                }
                for(int d = 0; d<D; d++){
```

```
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, R[d][j][l]);

model.addEq(constraintR,model.sum(BR[d][j][l],DR[d][j][l]) );
                        }
                    }
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, DR[d][j][l]);
                            model.addEq(constraintR,0 );
                        }
                    }
                }
            }
            if(step == 13 && improver >0){
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
```

```
    for(int j = 0; j<v;j++){
        for( int f = 0; f<F; f++){
        Dx[i][j][f] =  model.numVar(0, 1);
        }
    }
}
// defining L[v][f] as Double
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        L[j][f] =model.numVar(0,1);
    }
}
// defining C[v][f] as integer
for(int j = 0; j < v; j++){
    for(int f = 0 ; f< F; f++){
        C[j][f] = model.numVar(0,Double.MAX_VALUE);
    }
}
//defining V[v] as double
for(int j = 0 ; j< v ; j++){
    V[j] =  model.numVar(0,Double.MAX_VALUE);
}
//defining R[d][v][H] as double
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            R[d][j][l] =model.numVar(0,Double.MAX_VALUE);
        }
    }
```

```
                }
                for(int i = fixer;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, Dx[i][j][f]);
                            model.addEq(constraint312,0 );
                        }
                    }

                }
            }
            if(step == 13 && improver == 0){
                for(int i = 0;i<n;i++){
                    for(int j =0; j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        }
                    }
                }
                 for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Bx[i][j][f] =  model.boolVar();
                            }
```

```
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Dx[i][j][f]      =              model.numVar(0,
Double.MAX_VALUE);
                        }
                    }
                }

            //defining L[v][f]'s boolean part
            for(int j = 0; j <v; j++){
                for(int f = 0; f<F; f++){
                    BL[j][f] = model.boolVar();
                }
            }
            //defining L[v][f]'s double part
            for(int j = 0; j <v; j++){
                for(int f = 0; f<F; f++){
                    DL[j][f] = model.numVar(0,1);
                }
            }
            //defining V[v]'s boolean part
                    for(int j = 0 ; j< v ; j++){
                        BV[j] = model.boolVar();
                    }
                    //defining V[v]'s double part
                    for(int j = 0 ; j< v ; j++){
```

```
                    DV[j] =  model.numVar(0,1);
                }

            // defining R[d][v][H]'s boolean part
            for(int d = 0; d<D; d++){
                for(int j = 0 ; j <v; j++){
                    for(int l= 0; l< H; l++){
                        BR[d][j][l] = model.boolVar();
                    }
                }
            }
            //defining R[d][v][H]'s double part
            for(int d = 0; d<D; d++){
                for(int j = 0 ; j <v; j++){
                    for(int l= 0; l< H; l++){
                        DR[d][j][l] =model.numVar(0,1);
                    }
                }
            }
            for(int j=0;j<v;j++){
                for(int f=0;f<F;f++){
                    IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                    constraintL.addTerm(1.0, L[j][f]);

model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
                }
            }
            for(int j=0;j<v;j++){
                for(int f=0;f<F;f++){
```

```
                        IloLinearNumExpr                        constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, DL[j][f]);
                        model.addEq(constraintL,0);
                    }
                }
                for(int j=0;j<v;j++){
                        IloLinearNumExpr                        constraintV=
model.linearNumExpr();
                        constraintV.addTerm(1.0, V[j]);
                        model.addEq(constraintV,model.sum(BV[j],DV[j]));
                    }
                for(int j=0;j<v;j++){
                    IloLinearNumExpr constraintV= model.linearNumExpr();
                    constraintV.addTerm(1.0, DV[j]);
                    model.addEq(constraintV,0);
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, R[d][j][l]);

model.addEq(constraintR,model.sum(BR[d][j][l],DR[d][j][l]) );
                        }
                    }
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
```

```
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();

                            constraintR.addTerm(1.0, DR[d][j][l]);

                            model.addEq(constraintR,0 );

                        }

                    }

                }

            }

        /*  if( step == 14 && binarizer == 5){

                for(int i = 0;i<n;i++){

                    for(int j =0; j<v;j++){

                        for(int f=0;f<F;f++){

                            IloLinearNumExpr                constraint312=
model.linearNumExpr();

                            constraint312.addTerm(1.0, x[i][j][f]);

                            model.addEq(constraint312,fixedXValues[i][j][f
] );

                        }

                    }

                }

                 for(int i = 0; i<n; i++){

                        for(int j = 0; j<v;j++){

                            for( int f = 0; f<F; f++){

                            Bx[i][j][f] =  model.boolVar();

                            }

                        }

                    }

                    for(int i = 0; i<n; i++){

                        for(int j = 0; j<v;j++){

                            for( int f = 0; f<F; f++){
```

```
                    Dx[i][j][f]      =              model.numVar(0,
Double.MAX_VALUE);
                    }
                }
            }

        //defining L[v][f]'s boolean part
        for(int j = 0; j <v; j++){
            for(int f = 0; f<F; f++){
                BL[j][f] = model.boolVar();
            }
        }
        //defining L[v][f]'s double part
        for(int j = 0; j <v; j++){
            for(int f = 0; f<F; f++){
                DL[j][f] = model.numVar(0,1);
            }
        }
        //defining V[v]'s boolean part
                for(int j = 0 ; j< v ; j++){
                    BV[j] = model.boolVar();
                }
                //defining V[v]'s double part
                for(int j = 0 ; j< v ; j++){
                    DV[j] =  model.numVar(0,1);
                }

        // defining R[d][v][H]'s boolean part
        for(int d = 0; d<D; d++){
```

```
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            BR[d][j][l] = model.boolVar();
                        }
                    }
                }
                //defining R[d][v][H]'s double part
                for(int d = 0; d<D; d++){
                    for(int j = 0 ; j <v; j++){
                        for(int l= 0; l< H; l++){
                            DR[d][j][l] =model.numVar(0,1);
                        }
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                        constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, L[j][f]);
                        model.addEq(constraintL,model.sum(BL[j][f],DL[j][f
]));
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                        constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, DL[j][f]);
                        model.addEq(constraintL,0);
                    }
```

```
                }
                for(int j=0;j<v;j++){
                        IloLinearNumExpr                        constraintV=
model.linearNumExpr();
                        constraintV.addTerm(1.0, V[j]);
                        model.addEq(constraintV,model.sum(BV[j],DV[j]));
                    }
                for(int j=0;j<v;j++){
                    IloLinearNumExpr constraintV= model.linearNumExpr();
                    constraintV.addTerm(1.0, DV[j]);
                    model.addEq(constraintV,0);
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, R[d][j][l]);
                            model.addEq(constraintR,model.sum(BR[d][j][l],
DR[d][j][l]) );
                        }
                    }
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                    constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, DR[d][j][l]);
                            model.addEq(constraintR,0 );
                        }
```

```
                    }
                }
            }*/
        //  System.out.println(fixer);
            if(step==4){

                 for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Bx[i][j][f] =  model.boolVar();
                            }
                        }
                    }
                    for(int i = 0; i<n; i++){
                        for(int j = 0; j<v;j++){
                            for( int f = 0; f<F; f++){
                            Dx[i][j][f]      =             model.numVar(0,
Double.MAX_VALUE);
                            }
                        }
                    }

                //defining L[v][f]'s boolean part
                for(int j = 0; j <v; j++){
                    for(int f = 0; f<F; f++){
                        BL[j][f] = model.boolVar();
                    }
                }
                //defining L[v][f]'s double part
```

```
for(int j = 0; j <v; j++){
    for(int f = 0; f<F; f++){
        DL[j][f] = model.numVar(0,1);
    }
}
//defining V[v] as double
for(int j = 0 ; j< v ; j++){
    V[j] =  model.boolVar();
}
//defining V[v]'s boolean part
    /*  for(int j = 0 ; j< v ; j++){
            BV[j] = model.boolVar();
        }
        //defining V[v]'s double part
        for(int j = 0 ; j< v ; j++){
            DV[j] =  model.numVar(0,1);
        }*/

// defining R[d][v][H]'s boolean part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
        for(int l= 0; l< H; l++){
            BR[d][j][l] = model.boolVar();
        }
    }
}
//defining R[d][v][H]'s double part
for(int d = 0; d<D; d++){
    for(int j = 0 ; j <v; j++){
```

```
                        for(int l= 0; l< H; l++){
                            DR[d][j][l] =model.numVar(0,1);
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j=0;j<v;j++){
                        for(int f=0;f<F;f++){
                            IloLinearNumExpr                constraint312=
model.linearNumExpr();
                            constraint312.addTerm(1.0, x[i][j][f]);

model.addEq(constraint312,fixedXValues[i][j][f] );
                        }
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, L[j][f]);

model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                    constraintL=
model.linearNumExpr();
                        constraintL.addTerm(1.0, DL[j][f]);
                        model.addEq(constraintL,0);
```

```
                    }
                }
            /*  for(int j=0;j<v;j++){
                    IloLinearNumExpr                    constraintV=
model.linearNumExpr();
                    constraintV.addTerm(1.0, V[j]);
                    model.addEq(constraintV,model.sum(BV[j],DV[j]));
                }
                for(int j=0;j<v;j++){
                    IloLinearNumExpr constraintV= model.linearNumExpr();
                    constraintV.addTerm(1.0, DV[j]);
                    model.addEq(constraintV,0);
                }*/
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, R[d][j][l]);

model.addEq(constraintR,model.sum(BR[d][j][l],DR[d][j][l]) );
                        }
                    }
                }
                for(int d = 0; d<D; d++){
                    for(int j=0;j<v;j++){
                        for(int l=0;l<H;l++){
                            IloLinearNumExpr                constraintR=
model.linearNumExpr();
                            constraintR.addTerm(1.0, DR[d][j][l]);
                            model.addEq(constraintR,0 );
```

```
                    }
                }
            }

        }
    //  System.out.println("step : " +step);
        else if(step>0 && step<4){
             for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Bx[i][j][f] =  model.boolVar();
                        }
                    }
                }
                for(int i = 0; i<n; i++){
                    for(int j = 0; j<v;j++){
                        for( int f = 0; f<F; f++){
                        Dx[i][j][f]      =             model.numVar(0,
Double.MAX_VALUE);
                        }
                    }
                }
            //  System.out.println(fixer);
                for(int i = 0; i<fixer; i++){
                    for(int j=0;j<v;j++){
                        for(int f=0;f<F;f++){
                        IloLinearNumExpr                constraint312=
model.linearNumExpr();
                        constraint312.addTerm(1.0, x[i][j][f]);
```

```
model.addEq(constraint312,fixedXValues[i][j][f] );
                        }
                    }
                }


             }
            for(int i=fixer;i<n;i++){
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                    constraint30=
model.linearNumExpr();
                        constraint30.addTerm(1.0, x[i][j][f]);

model.addEq(constraint30,model.sum(Bx[i][j][f],Dx[i][j][f]) );
                    }
                }
            }


            for(int i=fixer;i<fixer+delta;i++){
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                    constraintb=
model.linearNumExpr();
                        constraintb.addTerm(1.0, Dx[i][j][f]);
                        model.addEq(constraintb, 0);
                    }
                }
            }
            for(int i=fixer+delta;i<n;i++){
                for(int j=0;j<v;j++){
```

```
                    for(int f=0;f<F;f++){
                        IloLinearNumExpr                        constraintI=
model.linearNumExpr();
                        constraintI.addTerm(1.0, Bx[i][j][f]);
                        model.addEq(constraintI, 0);


                    }
                }
            }
        /*  for(int j=0;j<v;j++){
                for(int f=0;f<F;f++){
                    IloLinearNumExpr constraintL= model.linearNumExpr();
                    constraintL.addTerm(1.0, L[j][f]);
                    model.addEq(constraintL,model.sum(BL[j][f],DL[j][f]));
                }
            }
            for(int j=0;j<v;j++){
                IloLinearNumExpr constraintV= model.linearNumExpr();
                constraintV.addTerm(1.0, V[j]);
                model.addEq(constraintV,model.sum(BV[j],DV[j]));
            }
            for(int d = 0; d<D; d++){
                for(int j=0;j<v;j++){
                    for(int l=0;l<H;l++){
                        IloLinearNumExpr                        constraintR=
model.linearNumExpr();
                        constraintR.addTerm(1.0, R[d][j][l]);
                        model.addEq(constraintR,model.sum(BR[d][j][l],DR[d
][j][l]) );
                    }
```

```
    }
}*/
//System.out.println(fixer);

//defining demand satisfaction constraint
for(int i =0; i<n; i++){
    IloLinearNumExpr constraint2 = model.linearNumExpr();
    for(int j=0;j<v;j++){
        for(int f=0;f<F;f++){
            constraint2.addTerm(1.0,x[i][j][f]);
        }
    }
    model.addEq(constraint2,1.0);
}//System.out.println(fixer);
for(int j=0;j<v;j++){
    for(int f=0;f<F;f++){
        IloLinearNumExpr constraint1 = model.linearNumExpr();
        for(int i =0; i<n;i++){
            constraint1.addTerm(1,x[i][j][f]);
        }
    model.addEq(constraint1, C[j][f]);
    }
}

//adding route distinguishing constraints
for(int j=0;j<v;j++){
    for(int f=0;f<F;f++){
        IloLinearNumExpr constraint3 = model.linearNumExpr();
        constraint3.addTerm(smax,L[j][f]);
```

```
            model.addGe(constraint3, C[j][f]);


        }
    }
    for(int j=0;j<v;j++){
        for(int f=0;f<F;f++){
            IloLinearNumExpr constraint4 = model.linearNumExpr();
            constraint4.addTerm(1,L[j][f]);
            model.addLe(constraint4, C[j][f]);


        }
    }
    //adding vehicle number constraint
    for(int j=0;j<v;j++){
        IloLinearNumExpr constraint5 = model.linearNumExpr();
        for(int i=0;i<n;i++){
        for(int f=0; f<F;f++){
            constraint5.addTerm(1.0,x[i][j][f]);
        }
        }
        model.addLe(constraint5, model.prod(m, V[j]));
    }
    //adding capacity constraint in normal frequencies
    for(int j=0;j<v;j++){
        for(int f=0;f<F;f++){
            IloLinearNumExpr constraint6 = model.linearNumExpr();
            for(int i=0;i<n;i++){
                constraint6.addTerm(y[i][f],x[i][j][f]);
            }
```

```
                    model.addLe(constraint6, c[j]);
                }
            }
            //adding capacity constraint in coinciding frequencies
            for(int j=0;j<v;j++){
                    IloLinearNumExpr constraint7 = model.linearNumExpr();
                    IloLinearNumExpr constraint8 = model.linearNumExpr();
                    IloLinearNumExpr constraint9 = model.linearNumExpr();
                    IloLinearNumExpr constraint10 = model.linearNumExpr();
                    IloLinearNumExpr constraint11= model.linearNumExpr();
                for(int i=0;i<n;i++){
                    for(int f=0;f<F;f++){
                     for(int f1=0;f1<5;f1++){
                                if(f==F1[f1]){
constraint7.addTerm(y[i][f],x[i][j][f]);
                                    }
                                 if(f==F2[f1]){
constraint8.addTerm(y[i][f],x[i][j][f]);
                                    }
                                if(f==F3[f1]){
constraint9.addTerm(y[i][f],x[i][j][f]);
                                    }
                            if(f==F4[f1]){
constraint10.addTerm(y[i][f],x[i][j][f]);
                                    }
                            if(f==F5[f1]){
```

```
constraint11.addTerm(y[i][f],x[i][j][f]);

                                }

                            }

                        }

                    }
                         model.addLe(constraint7, c[j]);
                         model.addLe(constraint8, c[j]);
                         model.addLe(constraint9, c[j]);
                         model.addLe(constraint10, c[j]);
                         model.addLe(constraint11, c[j]);

                    }


            //adding maximum customers to be visited
            for(int j=0;j<v;j++){
                    IloLinearNumExpr constraint12 = model.linearNumExpr();
                    IloLinearNumExpr constraint13= model.linearNumExpr();
                    IloLinearNumExpr constraint14 = model.linearNumExpr();
                    IloLinearNumExpr constraint15 = model.linearNumExpr();
                    IloLinearNumExpr constraint16 = model.linearNumExpr();
                for(int f=0;f<F;f++){
                        for(int f1=0;f1<5;f1++){
                                if(f==F1[f1]){
                                        constraint12.addTerm(1.0,C[j][f]);
                                    }
```

```
                                if(f==F2[f1]){
                                    constraint13.addTerm(1.0,C[j][f]);
                                }
                                if(f==F3[f1]){
                                    constraint14.addTerm(1.0,C[j][f]);
                                }
                                if(f==F4[f1]){
                                    constraint15.addTerm(1.0,C[j][f]);
                                }
                                if(f==F5[f1]){
                                    constraint16.addTerm(1.0,C[j][f]);
                                }
                            }
                        }
                        model.addLe(constraint12, smax);
                        model.addLe(constraint13, smax);
                        model.addLe(constraint14, smax);
                        model.addLe(constraint15, smax);
                        model.addLe(constraint16, smax);
                    }

                //adding routing rules
                for(int j=0;j<v;j++){
                    for(int i=0;i<n;i++){
                        for(int k=0;k<n;k++){
                            if(i<k){
                                IloLinearNumExpr constraint17 =
model.linearNumExpr();
                                IloLinearNumExpr constraint18 =
model.linearNumExpr();
```

```
                        IloLinearNumExpr        constraint19        =
model.linearNumExpr();
                        IloLinearNumExpr        constraint20        =
model.linearNumExpr();
                        IloLinearNumExpr        constraint21        =
model.linearNumExpr();
                    for(int f=0;f<F;f++){
                     for(int f1=0;f1<5;f1++){
                            if(f == F1[f1]){
                                constraint17.addTerm(1.0,x[i][j][f]);
                                constraint17.addTerm(1.0,x[k][j][f]);
                            }
                            if(f == F2[f1]){
                                constraint18.addTerm(1.0,x[i][j][f]);
                                constraint18.addTerm(1.0,x[k][j][f]);
                            }
                            if(f == F3[f1]){
                                constraint19.addTerm(1.0,x[i][j][f]);
                                constraint19.addTerm(1.0,x[k][j][f]);
                            }
                            if(f == F4[f1]){
                                constraint20.addTerm(1.0,x[i][j][f]);
                                constraint20.addTerm(1.0,x[k][j][f]);
                            }
                            if(f == F5[f1]){
                                constraint21.addTerm(1.0,x[i][j][f]);
                                constraint21.addTerm(1.0,x[k][j][f]);
                            }
                            }
                        }
```

```
                        model.addLe(constraint17, b[i][k]);
                        model.addLe(constraint18, b[i][k]);
                        model.addLe(constraint19, b[i][k]);
                        model.addLe(constraint20, b[i][k]);
                        model.addLe(constraint21, b[i][k]);
                        }

                    }
                }
            }
            //adding repetition omitting constraint
            for(int j=0;j<v;j++){
                for(int l=0;l<H;l++){
                                    IloLinearNumExpr    constraint10    =
model.linearNumExpr();
                                    IloLinearNumExpr    constraint11    =
model.linearNumExpr();
                                    IloLinearNumExpr    constraint12    =
model.linearNumExpr();
                                    IloLinearNumExpr    constraint13    =
model.linearNumExpr();
                                    IloLinearNumExpr    constraint14    =
model.linearNumExpr();
                                    constraint10.addTerm(1.0, L[j][1]);
                                    constraint10.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint10, R[0][j][l]);

                                    constraint11.addTerm(1.0, L[j][2]);
                                    constraint11.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint11, R[1][j][l]);
```

```
                                    constraint12.addTerm(1.0, L[j][3]);
                                    constraint12.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint12, R[2][j][l]);

                                    constraint13.addTerm(1.0, L[j][4]);
                                    constraint13.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint13, R[3][j][l]);

                                    constraint14.addTerm(1.0, L[j][5]);
                                    constraint14.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint14, R[4][j][l]);

                    }
                   }
               for(int j=0;j<v;j++){
                   for(int l=1;l<H;l =l+2){
                                    IloLinearNumExpr     constraint15     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint16     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint17     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint18     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint19     =
model.linearNumExpr();
                                    constraint15.addTerm(1.0, L[j][6]);
                                    constraint15.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint15, R[0][j][l]);

                                    constraint16.addTerm(1.0, L[j][7]);
```

```
                                    constraint16.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint16, R[1][j][l]);

                                    constraint17.addTerm(1.0, L[j][8]);
                                    constraint17.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint17, R[2][j][l]);

                                    constraint18.addTerm(1.0, L[j][9]);
                                    constraint18.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint18, R[3][j][l]);

                                    constraint19.addTerm(1.0, L[j][10]);
                                    constraint19.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint19, R[4][j][l]);

                    }
                   }
               for(int j=0;j<v;j++){
                   for(int l=2;l<51;l =l+3){
                                    IloLinearNumExpr     constraint20     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint21     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint22     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint23     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint24     =
model.linearNumExpr();
                                    constraint20.addTerm(1.0, L[j][11]);
                                    constraint20.addTerm(-1.0, L[j][0]);
```

```
                                    model.addLe(constraint20, R[0][j][l]);

                                    constraint21.addTerm(1.0, L[j][12]);
                                    constraint21.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint21, R[1][j][l]);

                                    constraint22.addTerm(1.0, L[j][13]);
                                    constraint22.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint22, R[2][j][l]);

                                    constraint23.addTerm(1.0, L[j][14]);
                                    constraint23.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint23, R[3][j][l]);

                                    constraint24.addTerm(1.0, L[j][15]);
                                    constraint24.addTerm(-1.0, L[j][0]);
                                    model.addLe(constraint24, R[4][j][l]);

                 }
                }
            for(int j=0;j<v;j++){
                for(int l=3;l<H;l =l+4){
                                    IloLinearNumExpr     constraint25     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint26     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint27     =
model.linearNumExpr();
                                    IloLinearNumExpr     constraint28     =
model.linearNumExpr();
```

```
                                    IloLinearNumExpr     constraint29     =
model.linearNumExpr();

                                    constraint25.addTerm(1.0, L[j][16]);

                                    constraint25.addTerm(-1.0, L[j][0]);

                                    model.addLe(constraint25, R[0][j][l]);


                                //
System.out.println("R[0]["+(j+1)+"]["+(l+1)+"]= " );

                                    constraint26.addTerm(1.0, L[j][17]);

                                    constraint26.addTerm(-1.0, L[j][0]);

                                    model.addLe(constraint26, R[1][j][l]);


                                    constraint27.addTerm(1.0, L[j][18]);

                                    constraint27.addTerm(-1.0, L[j][0]);

                                    model.addLe(constraint27, R[2][j][l]);


                                    constraint28.addTerm(1.0, L[j][19]);

                                    constraint28.addTerm(-1.0, L[j][0]);

                                    model.addLe(constraint28, R[3][j][l]);


                                    constraint29.addTerm(1.0, L[j][20]);

                                    constraint29.addTerm(-1.0, L[j][0]);

                                    model.addLe(constraint29, R[4][j][l]);


                        }

                    }


                if( model.solve()){

                    step++;
```

```
                //      System.out.println("objective["  +  (step)+  "]="  +
model.getValue(obj));

                   if(step<5){

                System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));




                double replenishmentCost = 0;

                double holdingCost = 0;

                for(int i=0; i<n;i++){

                    for(int j=0;j<v;j++){

                        for(int f=0;f<F;f++){

                            fixedXValues[i][j][f]                          =
model.getValue(x[i][j][f]);

                            replenishmentCost     =     replenishmentCost     +
K[i][f]*model.getValue(x[i][j][f]);

                            holdingCost    =    holdingCost    +    y[i][f]    *
0.5*model.getValue(x[i][j][f])*h1;

                           //                   System.out.println("x["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]=" + model.getValue(x[i][j][f]));

                           //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);

                           //  System.out.println("model is solved");




                        }

                    }

                }

            //       System.out.println("    replenishmentCost    =    "    +
replenishmentCost + " holdingCost = " + holdingCost);



                double repetitionCost = 0;

                double routeCost = 0;
```

```
                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        if(f==0){

                            repetitionCost = repetitionCost+
(g[j]*p[f]*model.getValue(L[j][f]));




                        }routeCost = routeCost+
(r[j]*p[f]*model.getValue(C[j][f]));

                //        System.out.println("C["+(j+1)+"]["+(f+1)+"]=    "
+model.getValue(C[j][f])); ;


                    }

                }//System.out.println("    repetitionCost    =    "    +
repetitionCost + " routeCost = " + routeCost);



                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                //        System.out.println("L["+(j+1)+"]["+(f+1)+"]= " +
model.getValue(L[j][f]));


                    }

                }

                double ownership = 0;

                for(int j=0;j<v;j++){

                System.out.println("V["+(j+1)+"]=          "          +
model.getValue(V[j]));

                     ownership = ownership + a[j] * model.getValue(V[j]);

                //  System.out.println(ownership);

                }

            //  System.out.println(" ownership = " + ownership );
```

```
                double rTotal =0;

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){

                            fixedRValues[d][j][l]                           =
model.getValue(R[d][j][l]);;

                            rTotal                =                 rTotal+
g[j]*model.getValue(R[d][j][l]);



                        }

                    }

                }

        //      System.out.println(" rTotal  = " + rTotal );



                double  totalCost  =  replenishmentCost  +  holdingCost  +
repetitionCost + routeCost + ownership + rTotal;

            //  System.out.println(" totalCost  = " + totalCost );

                  }

            }else{

                System.out.print(" no solution exists ");



            }

            if(step ==5){

                 System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));

                newobjective = model.getValue(obj);

            /*System.out.println("newobjective: " + newobjective);

                System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));*/

                for(int i=0; i<n;i++){
```

```
                    for(int j=0;j<v;j++){

                        for(int f=0;f<F;f++){

                            fixedXValues[i][j][f]                           =
model.getValue(x[i][j][f]);

                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);

                        }

                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        fixedCValues[j][f] = model.getValue(C[j][f]);



                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        fixedLValues[j][f] = model.getValue(L[j][f]);

                        System.out.println("L["+(j+1)+"]["+(f+1)+"]="     +
model.getValue(L[j][f]));

                    }

                }

                for(int j=0;j<v;j++){

                        fixedVValues[j] = model.getValue(V[j]);

                    //            System.out.println("V["+(j+1)+"]="      +
model.getValue(V[j]));

                    }

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){
```

```
                            fixedRValues[d][j][l]                           =
model.getValue(R[d][j][l]);

                        }

                    }

                }


                break;

            }

            if(step == 6){

                 System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));

            //  System.out.println("fixer : " + fixer);

                binarizer++;

                for(int i=0; i<n;i++){

                    for(int j=0;j<v;j++){

                        for(int f=0;f<F;f++){

                            fixedXValues[i][j][f]                           =
model.getValue(x[i][j][f]);

                            //System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);

                        }

                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        fixedCValues[j][f] = model.getValue(C[j][f]);



                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){
```

```
                        fixedLValues[j][f] = model.getValue(L[j][f]);

                        System.out.println("L["+(j+1)+"]["+(f+1)+"]="     +
model.getValue(L[j][f]));

                    }

                }

                for(int j=0;j<v;j++){

                        fixedVValues[j] = model.getValue(V[j]);

                    //            System.out.println("V["+(j+1)+"]="      +
model.getValue(V[j]));

                    }

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){

                            fixedRValues[d][j][l]                          =
model.getValue(R[d][j][l]);

                        }

                    }

                }

                double replenishmentCost = 0;

                double holdingCost = 0;

                for(int i=0; i<n;i++){

                    for(int j=0;j<v;j++){

                        for(int f=0;f<F;f++){

                            fixedXValues[i][j][f]                          =
model.getValue(x[i][j][f]);

                         replenishmentCost     =     replenishmentCost     +
K[i][f]*model.getValue(x[i][j][f]);

                         holdingCost    =    holdingCost    +    y[i][f]    *
0.5*model.getValue(x[i][j][f])*h1;

                        }

                    }

                }
```

```
                double repetitionCost = 0;

                double routeCost = 0;

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        if(f==0){

                            repetitionCost = repetitionCost+
(g[j]*p[f]*model.getValue(L[j][f]));




                        }routeCost = routeCost+
(r[j]*p[f]*model.getValue(C[j][f]));

                //        System.out.println("C["+(j+1)+"]["+(f+1)+"]=    "
+model.getValue(C[j][f])); ;



                    }

                }

                double ownership = 0;

                for(int j=0;j<v;j++){

                System.out.println("V["+(j+1)+"]=          "          +
model.getValue(V[j]));

                     ownership = ownership + a[j] * model.getValue(V[j]);

                //  System.out.println(ownership);

                }

            //  System.out.println(" ownership = " + ownership );



                double rTotal =0;

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){

                            fixedRValues[d][j][l]                    =
model.getValue(R[d][j][l]);;
```

```
                            rTotal                =                rTotal+
g[j]*model.getValue(R[d][j][l]);


                        }

                    }

                }

        //      System.out.println(" rTotal  = " + rTotal );


                double  totalCost  =  replenishmentCost  +  holdingCost  +
repetitionCost + routeCost + ownership + rTotal;

                System.out.println(" totalCost  = " + totalCost );

            }

            if(step == 7){

                 System.out.println("objective["   +   (step)+   "]="   +
model.getValue(obj));

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        fixedLValues[j][f] = model.getValue(L[j][f]);

                        System.out.println("L["+(j+1)+"]["+(f+1)+"]="     +
model.getValue(L[j][f]));

                    }

                }

                for(int i=0; i<n;i++){

                    for(int j=0;j<v;j++){

                        for(int f=0;f<F;f++){

                        //                  fixedXValues[i][j][f]        =
model.getValue(x[i][j][f]);

                            //System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);

                        }

                    }

                }
```

```
            //  System.out.println("fixer : " + fixer);
                if(newobjective> model.getValue(obj)){
                    improver++;
                newobjective = model.getValue(obj);
                for(int i=0; i<n;i++){
                    for(int j=0;j<v;j++){
                        for(int f=0;f<F;f++){
                            fixedXValues[i][j][f]                          =
model.getValue(x[i][j][f]);
                            //System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        fixedCValues[j][f] = model.getValue(C[j][f]);


                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        fixedLValues[j][f] = model.getValue(L[j][f]);
                        //
System.out.println("fixedLValues["+(j+1)+"]["+(f+1)+"]="                  +
fixedLValues[j][f]);
                    }
                }
                for(int j=0;j<v;j++){
                        fixedVValues[j] = model.getValue(V[j]);
```

```
                    //            System.out.println("V["+(j+1)+"]="      +
model.getValue(V[j]));

                    }

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){

                            fixedRValues[d][j][l]                          =
model.getValue(R[d][j][l]);

                        }

                    }

                }

            }else{

                improver = 0;

            }

            }

            if(step == 8){

                 System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));

                binarizer++;

                for(int i=0; i<n;i++){

                    for(int j=0;j<v;j++){

                        for(int f=0;f<F;f++){

                            fixedXValues[i][j][f]                          =
model.getValue(x[i][j][f]);

                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);

                        }

                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){
```

```
                        fixedCValues[j][f] = model.getValue(C[j][f]);


                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        fixedLValues[j][f] = model.getValue(L[j][f]);

                    //
System.out.println("fixedLValues["+(j+1)+"]["+(f+1)+"]="                    +
fixedLValues[j][f]);

                    }

                }

                for(int j=0;j<v;j++){

                        fixedVValues[j] = model.getValue(V[j]);

                    //            System.out.println("V["+(j+1)+"]="      +
model.getValue(V[j]));

                    }

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){

                            fixedRValues[d][j][l]                          =
model.getValue(R[d][j][l]);

                        }

                    }

                }

            }

            if(step == 9){

                 System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));

                newobjectivep = model.getValue(obj);

                if(newobjective>newobjectivep){
```

```
                //  System.out.println(model.getValue(obj));
                    improver++;
                    newobjective = model.getValue(obj);
                    for(int i=0; i<n;i++){
                        for(int j=0;j<v;j++){
                            for(int f=0;f<F;f++){
                                fixedXValues[i][j][f]                    =
model.getValue(x[i][j][f]);
                            //    System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                            }
                        }
                    }
                    for(int j=0;j<v;j++){
                        for(int f=0;f<F;f++){
                            fixedCValues[j][f] = model.getValue(C[j][f]);

                        }
                    }
                    for(int j=0;j<v;j++){
                        for(int f=0;f<F;f++){
                            fixedLValues[j][f] = model.getValue(L[j][f]);
                            //
System.out.println("fixedLValues["+(j+1)+"]["+(f+1)+"]="                +
fixedLValues[j][f]);
                        }
                    }
                    for(int j=0;j<v;j++){
                            fixedVValues[j] = model.getValue(V[j]);
                            //         System.out.println("V["+(j+1)+"]="     +
model.getValue(V[j]));
```

```
                        }
                    for(int d=0; d<D;d++){
                        for(int j=0;j<v;j++){
                            for(int l=0;l<H;l++){
                                fixedRValues[d][j][l]                       =
model.getValue(R[d][j][l]);
                            }
                        }
                    }
                }else{
                    improver = 0;
                }
            }
            if(step == 10){
                 System.out.println("objective["    +    (step)+    "]="    +
model.getValue(obj));
                binarizer++;
                for(int i=0; i<n;i++){
                    for(int j=0;j<v;j++){
                        for(int f=0;f<F;f++){
                            fixedXValues[i][j][f]                           =
model.getValue(x[i][j][f]);
                        //        System.out.println("fixedXValues["+(i+1)+
"]["+(j+1)+"]["+(f+1)+"]="  + fixedXValues[i][j][f]);
                        }
                    }
                }
                for(int j=0;j<v;j++){
                    for(int f=0;f<F;f++){
                        fixedCValues[j][f] = model.getValue(C[j][f]);
```

```
                    }

                }

                for(int j=0;j<v;j++){

                    for(int f=0;f<F;f++){

                        fixedLValues[j][f] = model.getValue(L[j][f]);

                    //
System.out.println("fixedLValues["+(j+1)+"]["+(f+1)+"]="                    +
fixedLValues[j][f]);

                    }

                }

                for(int j=0;j<v;j++){

                        fixedVValues[j] = model.getValue(V[j]);

                    //            System.out.println("V["+(j+1)+"]="      +
model.getValue(V[j]));

                    }

                for(int d=0; d<D;d++){

                    for(int j=0;j<v;j++){

                        for(int l=0;l<H;l++){

                            fixedRValues[d][j][l]                       =
model.getValue(R[d][j][l]);

                        }

                    }

                }
```

CURRICULUM VITAE

Name/Surname **: Niousha Karimi Dastjerd**

Nationality **: Iran**

Place and Date of Birth **: Urmia – 21.03.1988**

E-mail **: niousha.karimi@gmail.com**

EDUCATION STATUS:

- **Undergraduate** **:** 2009, Urmia University of Technology, Natural and Applied Scienses Faculty, Industrial Engineering

WORK EXPERIENCES AND AWARDS:

2014-2016	TOBB ETU	Full scholarship M.Sc. student

FOREIGN LANGUAGES:

Turkish

Persian

Azerbaijani

English

PUBLISHMENTS, PATENTS AND PRESENTATIONS ADOPTED FROM THESIS:

- **N. Karimi Dastjerd,** 2015. An strategic model and its analysis for the integrated fleet sizing and replenishment planning problem in vendor managed inventory systems, YAEM, September 9-11, Ankara, Turkey