COMPARISON OF INTEGER LINEAR PROGRAMMING AND DYNAMIC PROGRAMMING APPROACHES FOR ATM CASH REPLENISHMENT OPTIMIZATION PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FAZİLET ÖZER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2019

Approval of the thesis:

COMPARISON OF INTEGER LINEAR PROGRAMMING AND DYNAMIC PROGRAMMING APPROACHES FOR ATM CASH REPLENISHMENT OPTIMIZATION PROBLEM

submitted by **FAZİLET ÖZER** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** ________________

Prof. Dr. İsmail Hakkı Toroslu
Supervisor, **Computer Engineering Department** ________________

Prof. Dr. Pınar Karagöz
Co-supervisor, **Computer Engineering Department** ________________

Examining Committee Members:

Assoc. Prof. Dr. Lale Özkahya
Computer Engineering, Hacettepe Univ. ________________

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering, METU ________________

Assist. Prof. Dr. Ebru Aydın Göl
Computer Engineering, METU ________________

Date:

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

Name, Surname: FAZİLET ÖZER

Signature :

ABSTRACT

COMPARISON OF INTEGER LINEAR PROGRAMMING AND DYNAMIC PROGRAMMING APPROACHES FOR ATM CASH REPLENISHMENT OPTIMIZATION PROBLEM

ÖZER, FAZİLET
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. İsmail Hakkı Toroslu
Co-Supervisor : Prof. Dr. Pınar Karagöz

August 2019, 66 pages

In Automated Telling Machine (ATM) cash replenishment problem, banks aim to reduce the number of out-of-cash ATMs and duration of out-of-cash status. On the other hand, they want to reduce the cost of cash replenishment, as well. The problem conventionally involves forecasting ATM cash withdrawals, and then cash replenishment optimization on the basis of the forecast. We assume that reliable forecasts are already obtained for the amount of cash needed in ATMs. The focus of the thesis is cash replenishment optimization. After introducing Linear Programming based solutions, we propose a solution based on dynamic programming. Experiments conducted on real data reveal that the proposed approach can find the optimal solution more efficiently than linear programming.

Keywords: Cash Replenishment Problem, Replenishment Schedule, Optimization, Dynamic Programming, Linear Programming, Loading Cost, Interest Cost

ÖZ

ATM NAKİT YENİLEME OPTİMİZASYONU PROBLEMİ İÇİN TAMSAYILI DOĞRUSAL PROGRAMLAMA VE DİNAMİK PROGRAMLAMA YAKLAŞIMLARININ KARŞILAŞTIRILMASI

ÖZER, FAZİLET
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. İsmail Hakkı Toroslu
Ortak Tez Yöneticisi : Prof. Dr. Pınar Karagöz

Austos 2019 , 66 sayfa

Otomatik Anlatma Makinesi (ATM) nakit ikmali probleminde, bankalar, nakit dışı ATM sayısını ve nakit dışı durum süresini azaltmayı amaçlamaktadır. Öte yandan, nakit ikmal maliyetini de azaltmak istiyorlar. Sorun konvansiyonel olarak ATM nakit çekme tahminini ve ardından tahmin temelinde nakit yenileme optimizasyonunu içerir. ATM'lerde ihtiyaç duyulan nakit miktarı için zaten güvenilir tahminlerin alındığını varsayıyoruz. Tezin odak noktası nakit yenileme optimizasyonu. Doğrusal Programlama tabanlı çözümleri tanıttıktan sonra, dinamik programlamaya dayalı bir çözüm öneriyoruz. Gerçek veriler üzerinde yapılan deneyler, önerilen yaklaşımın lineer programlamaya göre en uygun çözümü bulabildiğini ortaya koymaktadır.

Anahtar Kelimeler: Otomatik Anlatma Makinesi, Nakit İkmali Problemi, Nakit Yenileme Optimizasyonu

To my dearest family and friends.

ACKNOWLEDGMENTS

I would like to express how thankful I am to work with my supervisor Prof. Dr. Ismail Hakki Toroslu. Thanks so much for his guidance for the last two years. Without his support I wouldn't go so far in this work.

I would like to express my gratitude to my co-supervisor Prof. Dr. Pnar Karagöz for her guidance for the last two years. Thanks for the help and support whenever I needed.

I would like to thank Safa for being there for me in my most stressful times during this process and for all of his support.

I would like to thank my friends for helping and encouraging me for my work. They tried to cheer me up when I needed it so deeply. And they encouraged me to finish this work in my more stressful times.

Finally, I would like to thank my family for all of their support. My dearest parents Semra and Ali, my brother Sabri, I couldn't go this far without your support. Thanks for everything.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ATM	Automated teller machines
ILP	Integer linear programming
DP	Dynammic Programming
MCP	Matrix chain product
ROP	Replenishment Optimization Problem
MIP	Mixed Integer Program

CHAPTER 1

INTRODUCTION

1.1 Motivation & Problem Definition

According to World Bank reports [1], the number of Automated Teller Machines (ATMs) all over the world increased by about 2.5 times within the last ten years. The increase in the use of ATMs facilitates banking services for both customer and banks, especially for simple and standard services such as cash withdrawal. Researches with real ATM investment data show that ATM usage is positively affecting the cost efficiency of the banks [2]. On the other hand, additional ATM management costs arise for banks [3] [4] [5]. One of the well-known ATM management problems is cash replenishment optimization, which mainly focuses on how often and how much cash to be loaded to an ATM in each cash replenishment period. The problem contains two optimization criteria. First of all, banks aim to reduce the amount of idle cash (i.e. cash that was loaded and was not withdrawn from ATM for a period of time), since this amount of cash cannot be utilized in a profitable way, thus it is considered as a loss. Therefore, it is aimed to avoid loading more amount of cash than needed. This cost is calculated as an interest lost in terms of the number of days cash stays in ATM idle [6] [7]. We call this cost as interest cost. On the other hand, loading a small amount of cash causes out-of-cash ATMs, and this is an important problem that affects customer satisfaction considerably. Additionally, cash replenishment incurs a cost involving cash transportation and loading process to an ATM [8] [9]. We call this cost as the loading cost. Hence, it is important to reduce the frequency of replenishment where possible. We call the total cost generated by interest and loading costs as replenishment cost. ATM replenishment optimization is based on keeping these factors balanced.

This problem can be divided into two steps: forecasting how much cash to be withdrawn each day, and finding an optimization algorithm for cash replenishment schedule. For the first step, we assume that a reliable forecast for the amount of cash to be withdrawn each day for a period of time (typically for a week) is available. There are several works focused on this first phase of the problem. Yeliz Ekinci, Jye-Chyi Lu, Ekrem Duman studied on forecasting the amount demand (Optimization of ATM cash replenishment with group-demand forecast) [10]. Also, Andrawis, R.R., Atiya, A.F., El-Shishiny, H., studied on the forecasting model which they considered various previous comparison studies and time series competitions as guidance in determining which individual forecasting models to test (for possible inclusion in the forecast combination system)[11]. Another study is conducted by Kalchschmidt, M., Verganti, R., Zotteri, G., which they aimed to examine the impact of heterogeneity of customer requests on demand forecasting approaches based on three action research cases in their study[12]. Lastly, Venkatesh, K., Ravi, V., Prinzie, A., den Poel, D.V., studied on cash demand forecasting in ATMs which they applied neural networks and clustering[13].

The focus of this work is on the second step. Given the reliable forecast, we propose a dynamic programming based solution for ATM cash replenishment, such that ATM is never out-of-cash, and the cost of replenishment and cash utilization is optimized. Assuming that maximum replenishment frequency is daily, loading only the required amount of daily cash does not create any interest cost while maximizing the loading cost. On the other hand, for the lowest cash loading frequency, such as weekly, the loading cost is minimized, but, the interest cost is maximized.

In this thesis, two solutions for the ATM cash replenishment problem is modeled and the result of those are compared. Also, the solutions are modeled for both a single ATM and grouped ATMs. Grouped ATMs means that group of ATMs very close to each other so that cash in all of them can be considered as a whole. Which means that, if at least one the ATMs have enough money, then there is no need to load money to others. In the scope of this thesis, two ATMs are considered for grouped ATMs and while determining the schedule of replenishment, starting and ending day of this period is considered to be same.

One solution is based on Linear Programming approach while the other one is based on Dynamic Programming approach. In the literature, there are mostly solutions using Linear Programming approach for this problem with different parameters. The parameters used in this model are the amount of money withdrawn in each day, interest rate, transportation cost, and location. There are two different models for this problem. One is modeled considering each ATMs separately and the other one considers location information as well. With this info, two ATMs close to each other is considered as one ATM, meaning that, not having money in one of these ATMs is not a problem as long as another one has enough money. So, there are two different models (for one ATM only and for two close ATMs) for both LP and DP models.

In the DP approach, the problem is modeled very similarly to the matrix chain multiplication problem such that n consecutive days ATM replenishment is modeled similarly to the multiplication of a sequence of n matrices. Moreover, dynamic programming based optimized replenishment is presented on a set of cases in comparison to baseline approaches. Those baseline approaches include the case which cash replenishment is made on every day and the one with one time in a week cash replenishment.

1.2 Cash Replenishment Optimization

Banks need to find out a way to optimize how much cash and how frequent to load cash into each ATM machine. Loading cash to an ATM has a cost independent from the amount loaded. We can reduce this cost by trying to reduce the number of replenishments. However, that means loading larger amounts each time an ATM is loaded, which generates an interest cost for each day that cash stays in the ATM. Therefore, an optimized solution tries to reduce the number of replenishment to decrease the loading cost and reduce the amount loaded into an ATM to reduce the interest cost. These two objectives are contradictory and therefore the optimum solution should do these decisions to minimize the overall cost.

The inputs of the cash replenishment optimization problem (ROP) are as follows:

predicted withdrawal amounts for N consecutive days for an ATM, *cash transportation/loading cost, and,*

the daily interest rate,

location (optional).

Location input is optional because problem is divided into two parts. First is for single ATM and the second one is for grouped ATMs. Grouped ATMs mean that they are too close to each other so that it is not a problem to not have enough money in one of the ATMs if the others have enough cash. In the scope of this thesis, only solutions for two grouped ATMs will be shown. As a result, both dynammic programming and integer linear programming approaches for single ATM and grouped ATMs will be examined in this thesis.

1.3 Contributions

- Linear programming approach for a single ATM

- Linear programming approach for grouped ATMS

- Dynamic programming approach for a single ATM

- Dynamic programming approach for grouped ATMS

Cash optimization problem is very famous in bank industry, but solutions are mainly focused on linear programming approach. However, linear programming may run very long especially with real life inputs. Furthermore, it can get longer when new parameters are added to this problem. We aimed to find another approach to compare the new one with linear programming and reduce the running time.

1.4 Organization of the Thesis

The organization of this thesis is as follows:

Chapter 2- Literature Survey contains the summary of previous studies contacted on the optimization of cash replenishment problem.

Chapter 3- Background explains background information on the research fields and methods used in this thesis in detail.

Chapter 4- Methods is the part where the detailed information on the collected dataset and the methods used are explained.

Chapter 5- Results and Discussions reveals the results of the experiments and discusses these results.

Chapter 6- Conclusion and Feature Works gives a summary and concludes thesis. In addition, future work ideas to enhance studies conducted in this thesis are given in this chapter.

CHAPTER 2

LITERATURE SURVEY

In the literature, there is a limited set of studies that are related to the ATM replenishment problem that we have introduced in this thesis. There is the Baumol model [14] which has been dominant for examining the demand for transactions at the micro level. Then, Miller and Orr [15] suggested a stochastic model. Also, Moraes and Nagano [16] proposed a policy for cash management by using the model Miller and Orr suggested. They do not identify a single ideal point for the cash balance, however an oscillation range between a lower limit, an ideal balance, and an upper limit.

There is another study on a probabilistic cash balance problem [17] in which they generated a linear programming model related to this problem. Furthermore, Elton and Gruber [5] suggested a dynamic programming model as well for the cash balance problem and stated that probabilistic changes in the cash level can be positive or negative. Another study [18] proposed a generic model of cash management which is viewed as an impulse control problem for a probabilistic money flow process. The study in [19] suggests a simple mathematical technique for cash management at the bank branches, especially both at the branch ATMs and at the cash desks. However, the work in [20] focuses on the problem of minimizing the expected time average cash balance that is dependent on the limitation in which the probability that all demands are met is at least some given number. The work in [21] focuses on the ATM network consisting of several banks' ATMs. Another work in [22], the linear programming approach is used for solving optimum cash replenishment routing problem of an ATM network. In [23], a mixed integer programming based approach is developed to solve the cash replenishment problem for a set of ATMs where cash is supplied from another set of cash centers. In [1], ATM withdrawal forecasts are used and a simulation-based optimization solution is developed for the cash replenishment decision.

There is a study conducted by (Bati & Gozupek) with ILP approach [24] which also focuses on solving both the routing and optimum replenishment of a set of ATMs. They also have a heuristic algorithm for their formulated ILP problem. Like this study, they also assume that the amount of the daily cash need for each ATM is predicted beforehand. In their model, lots of parameters are considered while formulating linear programming. Their optimization tries to find out a schedule which decides on which days the ATMs should be visited, how much amount of cash should be delivered to the ATMs, and what the route of the CIT(cash in transit) vehicles (vehicles that carry money to ATMs) should be in order to minimize the total cost. Furthermore, their problem is modeled for two types of ATM machines that are called as: 1) classical and 2) recycle ATMs. Classical ATMs have separate cassettes for deposit and cash withdrawal while recycle ATMs, also called as new-generation ATMs, have a single cassette for both operations.

There are few studies conducted on the routing of CIT vehicles, but mostly cash management is not a concern in those studies. For example, P. Kurdel and J. Sebestyenova describe the routing of CIT vehicles as some type of vehicle routing problem and use a genetic algorithm to refer it [25]. Moreover, D. J. du Toit, separates the routing of CIT vehicles and cash management problems, and also mostly works on the demand forecasting instead of optimization for cash management problem [26]. Opasanon and Lertsanti worked to solve some logistic problems for a company which main distribution center's relocation causes. In their study, they apply the analytic hierarchy process to evaluate and rank the importance of the logistics problems according to the requirements and needs of the companys policy makers [27]. H. Larrain, L. C. Coelho, and A. Cataldo study on joint vehicle routing and inventory management in ATM networks [28]. R. G. Van Anholt, L. C. Coelho, G. Laporte, and I. F. A. Vis focus on joint vehicle routing and inventory management of recirculation (recycle) ATMs [22].

Altunoglu, Castro, Simutis et al. studies on cash inventory management for out of working hours, during which replenishment of the ATMs is impossible. They suggest inventory models and policies under both perfect and imperfect information. Like our study, they used the forecast results to decide the replenishment policy which determines the number of days between two replenishments. Three different inventory policies were examined during the research; (M) policy with a lumpsum out of stock cost which is charged at the end of period, (t, M) policy with a lumpsum out of stock cost that is charged at time t within the period and (t, L, p) policy with a unit out of stock cost charged at time t within the period. Then, these three models are compared with the analysis of their performance according to headquarters full information optimal costs. Settings of fifteen problems with normally distributed demands are generated for the numerical analysis. Optimal solutions for each of these fifteen problems are searched by algorithms with three inventory policies.

Van Anholt, Coelho, Laporte, and Vis (2013) found out an solution for inventory-routing problem with pickups and deliveries for replenishing demands of the ATMs of a Dutch bank [22]. In their solution, they formulated the problem as a mixed-integer linear programming model, and suggested exactly an branch-and-cut algorithm for its resolution.

Chotayakul, Charnsetthikul, Pichitlamken, and Kobza (2013) determined how much money to place into ATMs and cash centers for each period of a given time [23]. Also, they modeled the problem as a Mixed Integer Program (MIP) and suggested an approach which is based on reformulating the model as a shortest path formulation in order to find out a near-optimal solution of the problem. These studies also assume the demand as given/known, i.e., it does not focus on demand forecasting part like other studies conducted before. The study conducted by Baker et al. (2013) might be the only study that focus on both the cash demand forecast and replenishment decision making. The forecasts are performed for each ATM in an isolation of the historical data from other ATMs even though this study confirmed the necessity of predicted cash demand usage.

There is another group working on the ATM cash replenishment problem and related this problem with the logistics problem.

Combination of goods is used in logistics very widely in distribution centers (Chen, Huang, Chen, & Wu, 2005)[29]. Retail orders are grouped and decisions of shipping are made based on these groupings instead of shipping directly from starting points to each retail store (Ballou, 1994). Ballou (1994) used zip codes for store aggregation[30]. Zarnani, Rahgozar, Lucas, and Taghiyareh (2009) focused on the usage of spatial clustering[31]. Daganzo (1984) formulated a vehicle routing technique, which is another version of the classical cluster-first, route-second approach[32]. First, the depot area is partitioned into districts containing clusters of stops and after that, vehicle route is built to serve each cluster.MichelVanderbeck (2012) and Gaur-Fisher (2004) also focused on similar studies[33].

None of these studies are the same as our problem, and, this might be the first introduction of the ATM cash replenishment optimization problem which tries to determine the optimum loading times for a given period for the given interest cost (obtained from the interest rate) and the fixed cash loading cost for each replenishment operation.

CHAPTER 3

BACKGROUND

In the scope of this thesis, two approaches are followed and applied to find a optimum solution for atm cash replenishment problem. The first one is integer linear programming approach and the second one is dynamic programming approach. In this section, I will give more detailed information about those approaches and how they are used for this specific solution.

3.1 Integer Linear Programming (ILP)

ILP is the name given to LP problems which have the additional constraint that some or all the variables have to be *integer*. It is very similar to linear programming except there is an extra constraint for it; variables must be integer.

3.1.1 Linear Programming (LP)

Linear programming, mathematical modeling method in which a linear function is maximized or minimized when subject to different limitations. This method was helpful in guiding quantitative choices in business planning, industrial engineering, and in the social and physical sciences to a lesser extent.

The solution of a linear programming problem increases to finding linear expression's optimum value which can be minimum or maximum, depending on the problem.

$f = c_1x_1 + ... + c_nx_n$

subject to a set of constraints expressed as inequalities:

$a_{11}x_1 + ... + a_{1n}x_n \leq b_1$

$a_{m1}x_m + ... + a_{mn}x_n \leq b_m \, \forall x_i \geq 0$

The as, bs, and cs are the constants specified by the problem's requirements and limitations. The basic assumption in applying this technique is that the different relationships between demand and availability are linear; meaning that, any of the xi is not raised to a power other than 1. In order to find a solution to this problem, it is a must to find the solution of the system of linear inequalities (meaning that, the set of n values of xi which simultaneously satisfies all of the inequalities). Then, the objective function is calculated by replacing the values of the variable xi in the equation which defines the function f.

Applications of the linear programming method were first seriously attempted in the late 1930s by the Soviet mathematician Leonid Kantorovich and the American economist Wassily Leontief in the fields of manufacturing schedules and economics, respectively, but their work was ignored for decades. Linear programming was widely used during World War II to cope with transportation, scheduling, and resource allocation subject to certain constraints such as cost and accessibility. These applications did much to establish the acceptability of this method, which gained further momentum in 1947 with the introduction of the simplex method by the American mathematician George Dantzig, which greatly simplified the solution of the problems of linear programming.

However, as progressively complicated issues involving more variables were tried, the amount of needed activities extended exponentially and surpassed the computing ability of even the most strong computers. After that, in 1979, Leonid Khachiyan - the Russian mathematician - found out a polynomial time algorithm in which the number of computational steps grows as a power of the number of variables instead of an exponential growth thus allowing the solution of previously inaccessible problems. On the other hand, Khachiyans algorithm which was called the ellipsoid method was slower than the simplex method when applied practically. Then, in 1984, Narendra Karmarkar - Indian mathematician - found out another polynomial time algorithm which is the interior point method and it is competitive with the simplex method.

3.2 Dynamic Programming (DP)

Dynamic Programming is a technique used for solving a complex problem by breaking it down into a set of simpler subproblems, solving each of those subproblems just once, and storing their solutions by using a memory-based data structure such as map, array. Each of these subproblems' solutions is indexed in some way, generally based on the values of the problem's input parameters in order to ease its lookup. So, when the next time, the same subproblem occurs, instead of recomputing its solution again, one simply looks up the previously computed one, and thus saves computing time. This technique is also called memorization.

Matrix chain multiplication problem is one of the famous problems solved by dynamic programming. The dynamic programming approach for cash replenishment optimization problem is based on matrix chain multiplication problem.

3.2.1 Matrix Chain Multiplication

One of the most well-known applications of DP method is for the matrix chain product problem [8]. Matrix chain multiplication problem - as the name implies - basically aims to find out the most efficient way of multiplying a sequence of matrices. In order to find the most efficient way of doing this operation, the order of the multiplications should be determined. Since, the matrix multiplication operation is associative the aim of the matrix chain product problem is to determine how to put parenthesis around the matrix pairs (input matrices or the ones obtained from previous multiplications) to execute the whole sequence of multiplication operation. Due to the associativity of the matrix multiplication operation, this parenthesization operations does not affect the result, but it affects the multiplication cost (i.e., the number of individual multiplications). Hence, how to place parenthesis must be found out in order to keep the multiplication cost at the minimum.

The recurrence relation of the matrix chain product (MCP) problem is given in Equation (1). In this formulation, the dimensions of matrix i are p_i and p_{i+1} and m[i, j] represents the minimum number of individual multiplications needed to multiply matrices i, (i+1), ...,j.

$$m[i,j] = min \begin{cases} min_{i \leq k < j}(m[i,k] + m[k+1,j] + p_{i-1}p_kp_j) \; if \; i < j \\ 0 \; if \; i = j \end{cases} \tag{31}$$

Matrix chain product problem resembles ATM replenishment optimization problem. While matrix chain product problem aims to determine the locations of parenthesis to minimize the total number of individual multiplications, ATM replenishment optimization problem aims to find out when and how much cash should be loaded to an ATM in order to minimize the total cost (transportation cost and interest cost). ROP problem has more parameters than MCP problem. Also, the cost calculation is more complicated and includes transportation cost and the interest cost.

Both optimization problems are defined between i to j (matrices from i to j or days from i to j). For both problems, all pairs of smaller instances between i and j must be explored to find out the optimal solution between i and j (that is by considering all instance pairs as i to k and k + 1 to j for all k values between i and j). However, MCP and ROP also have some differences:

- In MCP problem an instance with a single element (i.e. single matrix) has no (multiplication) cost. However, in ROP problem, an instance with a single element (i.e., replenishment period of a single day) has a transportation/loading cost. In MCP, combining two already solved (smaller) problem instances to form the solution of the larger problem instance has a cost, which corresponds to the cost of multiplying two matrices obtained from smaller problem instances. On the other hand, in ROP problem the cost of combining two smaller solutions to generate the solution of larger problem instance has no cost. That is, two consecutive replenishment periods can be combined to form a solution to a problem instances which starts at the first day of the first replenishment period and ends at the last day of the second replenishment period has no cost. In other words, in order to determine the cost of the solution from day i to day j, and, if the solutions for day i to k and day k + 1 to j have already been determined, just these two solutions can be added.

- In the MCP problem, only the solutions for smaller instances are needed to determine the solution of the larger problem instance. However, in ROP problem, in addition to the smaller problem instances, a special case corresponding to a single loading of the large problem instance should also be considered.

CHAPTER 4

METHODS

Banks need to find out a way to optimize how much cash and how frequent to load cash into each ATM machine. Loading cash to an ATM has a cost independent from the amount loaded. We can reduce this cost by trying to reduce the number of replenishments. However, that means loading larger amounts each time an ATM is loaded, which generates an interest cost for each day that cash stays in the ATM. Therefore, an optimized solution tries to reduce the number of replenishment to decrease the loading cost and reduce the amount loaded into an ATM to reduce the interest cost. These two objectives are contradictory and therefore the optimum solution should do these decisions to minimize the overall cost.

The inputs of the cash replenishment optimization problem (ROP) are as follows:

predicted withdrawal amounts for N consecutive days for an ATM,

cash transportation/loading cost, and,

the daily interest rate,

location (optional).

However, we can pre-calculate the total interest cost for the amount corresponding to each sub-period between the first day and the day N. This simplifies the ROP problem definition. We call the period between two consecutive replenishment days, including the former replenishment day and excluding the later one, as replenishment period. Below, we first define accumulated interest cost for all possible replenishment periods between day 1 and N, and, then use it to define the simpler version of the replenishment optimization problem.

4.1 Definition: Accumulated Interest Cost (I[i, j])

Total interest cost incurred for the amount corresponding to the predicted withdrawals of an ATM for a replenishment period [i, j] (i.e, from day i to day j) for a given daily interest rate. For a daily interest cost r, and predicted withdrawal amounts for all days between i and j ($i \leq k \leq j$) as $DailyAmount_k$ the accumulated interest cost I[i, j] is calculated as follows:

$\forall(i < k \leq j)$

$AmountWithInterest_k$ = $DailyAmount_k$ *$(1 + r)^{k-i}$

$InterestCost_k$ = $AmountWithInterest_k \;\; DailyAmount_k$

There will be no interest charged for the day i, and that amount is expected to be withdrawn in that day. The amount for day i+1 incurs 1-day interest, and, day i+2 incurs 2-day interest etc.

Also, please notice that the size of the accumulated interest cost matrix is (n-1) x n when n represents the number of days.

Consider a simple instance of ATM Replenishment Problem for 5 days with the following inputs:

Number of days: N = 5

Amount per days: [100, 200, 100, 300, 100]

Interest rate: r = 0.01 (i.e., 1% per day)

For this example, accumulated interest costs are calculated in three steps as follows:

4.1.1 Step 1: Calculate the total amount of money

For days from *1 to n* under the given interest rate *r*. In Table 4.1, the rows correspond to the days from 2 to 5, and the columns corresponds to the days from 0 to 5 where interest can be applied. An entry at row *i* and column *j* corresponds to the amount the money of day *i* will become with the given interest rate *r* in *j* days. The entries that are not calculated left empty. For example, for the day 2, the cash can be in the ATM at most for 1 day, if it is loaded at day 1.

Table 4.1: Accumulated Interests

Amount with interest	**Day 0**	**Day 1**	**Day 2**	**Day 3**	**Day 4**
Day 2 (200)	200	202			
Day 3 (100)	100	101	102.01		
Day 4 (300)	300	303	306.03	309.09	
Day 5 (100)	100	101	102.01	103.03	104.06

4.1.2 Step 2: Calculate the interest cost

If we extract the amount, then we will find the actual interest cost. Table 4.2 contains the interest cost for the amount of each day for the required days.

Table 4.2: Interest Costs

Interest cost	**Day 0**	**Day 1**	**Day 2**	**Day 3**	**Day 4**
Day 2 (200)	0	2			
Day 3 (100)	0	1	2.01		
Day 4 (300)	0	3	6.03	9.09	
Day 5 (100)	0	1	2.01	3.03	4.06

4.1.3 Step 3: Calculate the accumulated interest cost

When we load the cash at day 1 and then the next loading is at day 4, that means we need to load 3 days required cash at day 1 (i.e., for days 1, 2 and 3). Thus, for day 2 we will pay an interest cost for 1 day and for day 3 we will pay interest cost for 2 days, For this reason we need to calculate the accumulated interest costs for loading the cash at day *i* until day *j*. That means the next loading is on day *j + 1*. In Table 4.3 the rows correspond to loading days, and the columns correspond to the day until which the loading is done.

Table 4.3: Accumulated Interest Costs

I	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Day 1		2	4.01	13.10	17.16
Day 2			1	7.03	10.06
Day 3				3	5.01
Day 4					1

4.1.4 Definition: Replenishment Optimization Problem (ROP = (N, A[1..N], α)

For an ATM with predicted withdrawal amounts A[1] to A[N] for N consecutive days and a constant transportation cost α, ROP determines $k_1, k_2, .., k_m$ for cash replenishment days such that $k_1 = 1$, $\forall(i > 1)k_i \leq N$, and sum of the accumulated interest costs and the transfer costs for each replenishment period $[k_1, k_{2-1}]$, $[k_2, k3 - 1]$, ..., $[k_m, N]$ is minimized.

4.2 Integer linear programming (ILP) Modeling of ATM Cash Replenishment Optimization Problem

ROP is a typical discrete optimization problem, and thus, can be modeled as (mixed) integer linear programming (ILP) problem. Below we provide ILP modeling of ROP.

4.2.1 ILP Modeling of the Problem for a Single ATM

The parameters of ILP version of ROP are as follows:

D = number of days in the schedule

x_{ij} = *decision variable that corresponds to whether there is cash replenishment at day i for the ATM until day j (1 or 0)*

$I[i, j]$ = *accumulated interest cost*

α = *transportation cost for cash replenishment*

ROP tries to minimize the summation of transportation costs and accumulated interest costs (for replenishment periods between two consecutive replenishment days) by choosing the replenishment days between 1 and D. Therefore, the following formula considers all possible replenishment periods between day 1 and day N, and the decision variables (x_{ij}) will be chosen such that no overlapping replenishment periods can be selected and all days between 1 and D are included in the chosen replenishment periods.

Equation (43) guarantees that there will be exactly one replenishment period including day 1. Equation (44) means that if the first replenishment period was only 1 day, the second one should start from day 2. If the first replenishment period ends at a later day, there cannot be any replenishment period starting at day 2. In that case, all decision variables of equation (44) is zero. Again, the equation (44) enforces that if the first replenishment period was only one day (that is, x_{11} is 1), then, there must be one decision variable (such as x_{2j}) which is also 1. Similarly, following equations guarantees that every day between day 1 and day D are included in exactly one replenishment period.

minimize:

$$\sum_{i=1}^{D}\sum_{j=1}^{D}(x_{ij})(I[i, j] + c) \tag{41}$$

In order to find the optimized cost, the total cost must be minimized in this approach.

Above equation gives the total cost, and accumulated interest cost matrix is used as an input in this equation. To calculate the optimum cost, the corresponding cell from this matrix (I[i,j]) and transportation cost are summed up and multiplied with the value xij which is 0 or 1 and refers to the decision of whether there is cash replenishment to the ATM between the days i and j.

subject to:

$$0 \leq x_{ij} \leq 1, integer \tag{42}$$

$$\sum_{j=1}^{D}(x_{1j}) = 1 \tag{43}$$

$$\sum_{j=2}^{D}(x_{2j}) - \sum_{i=1}^{1}(x_{i1}) = 0 \tag{44}$$

$$\sum_{j=3}^{D}(x_{3j}) - \sum_{i=1}^{2}(x_{i2}) = 0 \tag{45}$$

$$\sum_{j=4}^{D}(x_{4j}) - \sum_{i=1}^{3}(x_{i3}) = 0 \tag{46}$$

...

$$\sum_{j=J}^{D}(x_{Jj}) - \sum_{i=1}^{(J-1)}(x_{i(J-1)}) = 0 \tag{47}$$

...

$$\sum_{j=(D-1)}^{D}(x_{(D-1)j}) - \sum_{i=1}^{(D-2)}(x_{i(D-2)}) = 0 \tag{48}$$

$$(-1) * \sum_{i=1}^{D} (x_{iD}) = -1 \tag{49}$$

While finding the minimized value, the above equations must be considered as constraints. The first equation is there because x_{ij} must be either 0 or 1. The second one is to make sure the replenishment schedule starts on the first day and the last one is to be sure it ends on the last day. The equations between those equations ensure whether some x_{ij} values are 1.

Illustration of the LP Solution for a Single ATM on a Case

Consider the same problem instance that we have introduced above, which is about an ATM Replenishment Problem for 5 days with the following parameters:

Number of days : n = 5

Amount per days: [100, 200, 100, 300, 100]

Interest rate: r = 0.01 (i.e., 1% per day)

Loading cost: α *= 5*

The decision variables correspond to all replenishment periods between day 1 and day 5. For example x_{24} represents the period from day 2 to day 4. That means three days of withdrawal amount (day 2, 3, and 4) is loaded at day 2, generating accumulated interest cost of I(2,4). This cost was previously calculated as 7.03 n previous section.Transportation/loading cost value 5 must be added to this value, making it 12.03.In below formula we use only rounded integer values. Therefore the variable $_{x24}$ is multiplied with 12. As the result of the minimization if $_{x24}$ is chosen to be 1, then it contributes total amount of 12 to the final cost.

The aim of this problem is to minimize z:

5*x_{11} + 5*x_{22} + 5*x_{33} + 5*x_{44} + 5*x_{55} + 7*x_{12} + 9*x_{13} + 18*x_{14} + 22*x_{15} + 5*x_{23}+ 12*x_{24} + 15*x_{25} + 8*x_{34} + 10*x_{35} + 6*x_{45}

subject to the following constraints:

$$0 \leq x_{11} \leq 1 \quad 0 \leq x_{12} \leq 1 \quad 0 \leq x_{13} \leq 1$$
$$0 \leq x_{14} \leq 1 \quad 0 \leq x_{15} \leq 1 \quad 0 \leq x_{22} \leq 1$$
$$0 \leq x_{23} \leq 1 \quad 0 \leq x_{24} \leq 1 \quad 0 \leq x_{25} \leq 1$$
$$0 \leq x_{33} \leq 1 \quad 0 \leq x_{34} \leq 1 \quad 0 \leq x_{35} \leq 1$$
$$0 \leq x_{44} \leq 1 \quad 0 \leq x_{45} \leq 1 \quad 0 \leq x_{55} \leq 1$$

$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 1$

$x_{22} + x_{23} + x_{24} + x_{25} - x_{11} = 0$

$x_{33} + x_{34} + x_{35} - x_{22} - x_{12} = 0$

$x_{44} + x_{45} - x_{33} - x_{13} - x_{23} = 0$

$-x_{15} - x_{25} - x_{35} - x_{45} - x_{55} = -1$

Solution is x_{13} and x_{45} , which corresponds to loading on the first day for amount up until the end of third day, and loading on the fourth day for the last two days together. This solution produces 15 as the total cost.

4.2.2 ILP Modeling of the Problem for Grouped ATMs

If two ATMs are close enough to each other, then banks may consider one ATMs not having money is negligible if another does have enough money. By this, the cost can be minimized. Furthermore, the transportation cost for the two ATMs is cheaper than one ATMs. With all that, there is an extra parameter added to the equations built in the first section, whether cash replenishment should be done separately or together. This changes the equation to be able to find the minimized total cost and constraints to satisfy this equation.

n = decision variable that corresponds to whether there is cash replenishment together or separately to the ATMs (1 means first ATM's cash replenishment, 2 means first ATM's cash replenishment and 3 means both ATMs' cash replenishment)

D = number of days in schedule

x_{ijn} *= decision variable that corresponds to whether there is cash replenishment at day i for the ATM until day j (1 or 0)*

In[i, j] = accumulated interest cost

α *= transportation cost for cash replenishment*

minimize:

$$\sum_{n=1}^{N}\sum_{i=1}^{D}\sum_{j=1}^{D}(x_{ijn})(I_n[i,j]+c_n) \tag{410}$$

The equation must cover all the cases while finding the minimum cost. Therefore, another parameter named n is added to the sum equation. The value for n starts from 1 and ends with 3. Value of n being 1 means cash replenishment of the first ATM, 2 means second ATM and 3 means both ATMs cash replenishment. Also, the accumulated interest rate is calculated for two ATMs by simply adding two separate matrices. Notice that, the value of c (transportation cost) changes with the value of n.

subject to:

$$0 \le x_{ijn} \le 1, integer \tag{411}$$

$$\sum_{j=1}^{D}(x_{1j1}) + \sum_{j=1}^{D}(x_{1j3}) = 1 \tag{412}$$

$$\sum_{j=1}^{D}(x_{1j2}) + \sum_{j=1}^{D}(x_{1j3}) = 1 \tag{413}$$

$$\sum_{j=2}^{D}(x_{2j1}) + \sum_{j=2}^{D}(x_{2j3}) - \sum_{i=1}^{1}(x_{i11}) - \sum_{i=1}^{1}(x_{i13}) = 0 \tag{414}$$

$$\sum_{j=2}^{D}(x_{2j2}) + \sum_{j=2}^{D}(x_{2j3}) - \sum_{i=1}^{1}(x_{i12}) - \sum_{i=1}^{1}(x_{i13}) = 0 \tag{415}$$

$$\sum_{j=3}^{D}(x_{3j1}) + \sum_{j=3}^{D}(x_{3j3}) - \sum_{i=1}^{2}(x_{i21}) - \sum_{i=1}^{2}(x_{i23}) = 0 \tag{416}$$

$$\sum_{j=3}^{D}(x_{3j2}) + \sum_{j=3}^{D}(x_{3j3}) - \sum_{i=1}^{2}(x_{i22}) - \sum_{i=1}^{2}(x_{i23}) = 0 \tag{417}$$

$$\sum_{j=4}^{D}(x_{4j1}) + \sum_{j=4}^{D}(x_{4j3}) - \sum_{i=1}^{3}(x_{i31}) - \sum_{i=1}^{3}(x_{i33}) = 0 \tag{418}$$

$$\sum_{j=4}^{D}(x_{4j2}) + \sum_{j=4}^{D}(x_{4j3}) - \sum_{i=1}^{3}(x_{i32}) - \sum_{i=1}^{3}(x_{i33}) = 0 \tag{419}$$

...

$$\sum_{j=J}^{D}(x_{Jj1}) + \sum_{j=J}^{D}(x_{Jj3}) - \sum_{i=1}^{(J-1)}(x_{i(J-1)1}) - \sum_{i=1}^{(J-1)}(x_{i(J-1)3}) = 0 \tag{420}$$

$$\sum_{j=J}^{D}(x_{Jj2}) + \sum_{j=J}^{D}(x_{Jj3}) - \sum_{i=1}^{(J-1)}(x_{i(J-1)2}) - \sum_{i=1}^{(J-1)}(x_{i(J-1)3}) = 0 \tag{421}$$

...

$$\sum_{j=(D-1)}^{D}(x_{(D-1)j1}) + \sum_{j=(D-1)}^{D}(x_{(D-1)j3}) - \sum_{i=1}^{(D-2)}(x_{i(D-2)1}) - \sum_{i=1}^{(D-2)}(x_{i(D-2)3}) = 0 \tag{422}$$

$$\sum_{j=(D-1)}^{D}(x_{(D-1)j2}) + \sum_{j=(D-1)}^{D}(x_{(D-1)j3}) - \sum_{i=1}^{(D-2)}(x_{i(D-2)2}) - \sum_{i=1}^{(D-2)}(x_{i(D-2)3}) = 0 \tag{423}$$

$$(-1) * \sum_{i=1}^{D}(x_{iD1}) - \sum_{i=1}^{D}(x_{iD3}) = -1 \tag{424}$$

$$(-1) * \sum_{i=1}^{D}(x_{iD2}) - \sum_{i=1}^{D}(x_{iD3}) = -1 \tag{425}$$

As before, the first equation is to ensure that x_{ijn} must be either 0 or 1. Two equations after that are to make sure the replenishment schedule starts on the first day and either cash is replenished separately or together. The last one is to be sure it ends on the last day and whether it should be done together or separately. The equations between those equations ensure whether some x_{ijn} values are 1.

Illustration of the LP Solution for Grouped ATMs on a Case

Consider the same problem instance that we have introduced above, which is about an ATM Replenishment Problem of two ATMs for 5 days with the following parameters:

Number of days : n = 5

Amount per days (first ATM): [100,200,100,300,100]

Amount per days (second ATM): [100,200,300,400,100]

Interest rate: r = 0.01 (i.e., 1% per day)

Loading cost (shared): α *= 5*

Loading cost (separate): α *= 8*

The aim of this problem is to minimize z:

$$
\begin{aligned}
5.0*x_{111}+7.0*x_{121}+9.01*x_{131}+18.1*x_{141}+22.16*x_{151}+5.0*x_{221}+\\
6.0*x_{231}+12.03*x_{241}+15.06*x_{251}+5.0*x_{331}+8.0*x_{341}+10.01*x_{351}+\\
5.0*x_{441}+6.0*x_{451}+5.0*x_{551}+\\
5.0*x_{112}+7.0*x_{122}+13.03*x_{132}+25.15*x_{142}+29.21*x_{152}+5.0*x_{222}+\\
8.0*x_{232}+16.04*x_{242}+19.07*x_{252}+5.0*x_{332}+9.0*x_{342}+11.01*x_{352}+\\
5.0*x_{442}+6.0*x_{452}+5.0*x_{552}+\\
8.0*x_{113}+12.0*x_{123}+20.04*x_{133}+\\
41.25*x_{143}+49.37*x_{153}+8.0*x_{223}+12.0*x_{233}+26.07*x_{243}+32.13*x_{253}+\\
8.0*x_{333}+15.0*x_{343}+19.02*x_{353}+8.0*x_{443}+10.0*x_{453}+8.0*x_{553}
\end{aligned}
\tag{426}
$$

subject to the following constraints:

$$
\begin{array}{ccccc}
0 \le x_{111} \le 1 & 0 \le x_{112} \le 1 & 0 \le x_{113} \le 1 & 0 \le x_{121} \le 1 & 0 \le x_{122} \le 1 \\
0 \le x_{123} \le 1 & 0 \le x_{131} \le 1 & 0 \le x_{132} \le 1 & 0 \le x_{133} \le 1 & 0 \le x_{141} \le 1 \\
0 \le x_{142} \le 1 & 0 \le x_{143} \le 1 & 0 \le x_{151} \le 1 & 0 \le x_{152} \le 1 & 0 \le x_{153} \le 1 \\
0 \le x_{221} \le 1 & 0 \le x_{222} \le 1 & 0 \le x_{223} \le 1 & 0 \le x_{231} \le 1 & 0 \le x_{232} \le 1 \\
0 \le x_{233} \le 1 & 0 \le x_{241} \le 1 & 0 \le x_{242} \le 1 & 0 \le x_{243} \le 1 & 0 \le x_{251} \le 1 \\
0 \le x_{252} \le 1 & 0 \le x_{253} \le 1 & 0 \le x_{331} \le 1 & 0 \le x_{332} \le 1 & 0 \le x_{333} \le 1 \\
0 \le x_{341} \le 1 & 0 \le x_{342} \le 1 & 0 \le x_{343} \le 1 & 0 \le x_{351} \le 1 & 0 \le x_{352} \le 1 \\
0 \le x_{353} \le 1 & 0 \le x_{441} \le 1 & 0 \le x_{442} \le 1 & 0 \le x_{443} \le 1 & 0 \le x_{451} \le 1 \\
0 \le x_{452} \le 1 & 0 \le x_{453} \le 1 & 0 \le x_{551} \le 1 & 0 \le x_{552} \le 1 & 0 \le x_{553} \le 1
\end{array}
$$

$$
x_{111}+x_{113}+x_{121}+x_{123}+x_{131}+x_{133}+x_{141}+x_{143}+x_{151}+x_{153}=1 \tag{427}
$$

$$x_{112} + x_{113} + x_{122} + x_{123} + x_{132} + x_{133} + x_{142} + x_{143} + x_{152} + x_{153} = 1 \quad (428)$$

$$x_{221} + x_{231} + x_{241} + x_{251} - x_{111} + x_{223} + x_{233} + x_{243} + x_{253} - x_{113} = 0 \quad (429)$$

$$x_{222} + x_{232} + x_{242} + x_{252} - x_{112} + x_{223} + x_{233} + x_{243} + x_{253} - x_{113} = 0 \quad (430)$$

$$x_{331} + x_{341} + x_{351} - x_{221} - x_{121} + x_{333} + x_{343} + x_{353} - x_{223} - x_{123} = 0 \quad (431)$$

$$x_{332} + x_{342} + x_{352} - x_{222} - x_{122} + x_{333} + x_{343} + x_{353} - x_{223} - x_{123} = 0 \quad (432)$$

$$x_{441} + x_{451} - x_{331} - x_{131} - x_{231} + x_{443} + x_{453} - x_{333} - x_{133} - x_{233} = 0 \quad (433)$$

$$x_{442} + x_{452} - x_{332} - x_{132} - x_{232} + x_{443} + x_{453} - x_{333} - x_{133} - x_{233} = 0 \quad (434)$$

$$-x_{151} - x_{251} - x_{351} - x_{451} - x_{551} - x_{153} - x_{253} - x_{353} - x_{453} - x_{553} = -1 \quad (435)$$

$$-x_{152} - x_{252} - x_{352} - x_{452} - x_{552} - x_{153} - x_{253} - x_{353} - x_{453} - x_{553} = -1 \quad (436)$$

Here in the solution, x_{ijk} means whether cash replenishment should be done between the days i and j, and k means whether it should be done together or separately (k=1 means first ATM's cash replenishment, k=2 means second ATM's cash replenishment and k=3 means together). x_{ijk} can be 0 or 1. First constraint is determined to be sure that there should be cash replenishment starting from the first day. Therefore, sum of all variables x_{1jk} should be equal to 1. Second constraint is determined to be sure that cash replenishment should end in the fifth day. Thus, at least one of the variables x_{i5k} should be one. Other three constraints are determined to be sure that there is a corresponding 1 for each x_{ijk} and thus, the sums should be equal to 0. As a result, optimized cost is found as 30 and x_{122}, x_{332} and x_{452} found as 1 which means that replenishment should be done together between the days 1-2, 3 and 4-5.

4.3 Dynamic programming (DP) Modeling of ATM Cash Replenishment Optimization Problem

In addition to using ILP, many discrete optimization problems are also solved by using dynamic programming (DP) approach. If a problem can be divided into (potentially overlapping) subproblems and the results of these subproblems can be combined to generate the result of the original-larger problem instance, then, typically DP can be used. In many cases (depending on problem structure and/or parameters) DP can be much more efficient than any other alternative. That is why DP is used to solve many discrete optimization problems.

4.3.1 Dynamic Programming based Modeling for a Single ATM

Using the above (simplified) definition of the ROP problem, which uses the accumulated interest cost (I [i, j] matrix for every pair of days from i to j) and a fixed loading cost α, the recurrence relation of ROP can be defined as follows:

$$c[i,j] = min \begin{cases} min_{i \le k < j}(c[i,k] + c[k+1,j]) \\ \sum_{r=i}^{j} I[i,j] + \alpha \end{cases} \tag{437}$$

In this definition, c[i, j] is the minimized replenishment cost for an ATM from day i to j (including j). As it is done for the MCP problem, the actual replenishment days producing this minimum cost can easily be obtained by storing the k values during the calculation of the cost matrix.

4.3.2 Illustration of the DP Solution for a Single ATM on a Case

In order to illustrate the DP method, we use the accumulated interest costs in Table 3. The calculation of the optimized cost (c[i, j]) values for the recurrence relation c is done for this example, and it is shown in Table 4.4.

Similar to the MCP, the entry at row 1 and column 5 (marked with ?) corresponds to the ROP problem instance that we want to solve. Aslo, as in MCP problem, each entry c[i, j] in ROP has a value corresponding to the optimum solution of the problem from day i to day j.

In Table 4.4, the entry c[1,2] shows the minimum cost of cash replenishment from day 1 to day 2. In order to find the minimum cost for this entry two alternatives must be compared: loading cash to an ATM day by day or loading it at once. The day by day replenishment cost which corresponds to the sum of c[1,1] and c[2,2]:

$$daybyday[1, 2] = cost[1, 1] + cost[2, 2] \tag{438}$$

On the other hand, the total replenishment cost for loading cash at once can be calculated as the sum of the accumulated interest cost from day 1 to 2 and a single loading cost:

$$atonce[1, 2] = \alpha + I[1, 2] \tag{439}$$

For this example, the day by day choice cost is 10 (=5+5), and at once choice cost is 7 (=5+2). Therefore, the value of c[1, 2] is 7. The calculations of the values of other entries of c matrix is done in the same way as the MCP in the order of diagonals.

The final entry to be calculated is c[1,5], which is the result of this problem instance. Calculations for c[1,5] requires choosing the minimum of the results of Equations 4.40-4.44.

$$(12345) = \alpha + I[1,5] = 5 + 17.25 = 22.25 \tag{440}$$

$$(1)(2345) = c[1,1] + c[2,5] = 5 + 12 = 17 \tag{441}$$

$$(12)(345) = c[1,2] + c[3,5] = 7 + 10.01 = 17.01 \tag{442}$$

$$(123)(45) = c[1,3] + c[4,5] = 9.1 + 6 = 15.1 \tag{443}$$

$$(1234)(5) = c[1,4] + c[5,5] = 14.1 + 5 = 19.1 \tag{444}$$

Among these 5 choices, the solution that corresponds to (123)(45) is the optimum one for c[1,5] whose value is 15.1. This solution means that we should load the first three days amount on day 1, and the last two days amount on day 4.

Table 4.4: Optimized Costs

c	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Day 1	5	7	9.1	14.1	?
Day 2		5	6	11	12
Day 3			5	8	10.01
Day 4				5	6
Day 5					5

4.3.3 Dynamic Programming based Modeling for Grouped ATMs

For grouped ATMs, cash replenishment can be done either together or separately to the ATMs. In the scope of this thesis, two ATMs are considered while talking about grouped ATMs. So, there is another parameter is added to the problem while considering two ATMs. Below is the recurrence relation for grouped ATMs:

$$c_2[i,j] = min \begin{cases} min_{i \le k < j}(c_2[i,k] + c_2[k+1,j]) \\ c_1^1[i,j] + c_1^2[i,j] \\ I_1^1[i,j] + I_1^2[i,j] + \beta \end{cases} \tag{445}$$

As seen, the summation of already calculated two separate ATMs is added to recurrence relation. c_1^1 and c_1^2 matrices are the calculated optimized cost matrices for the first and second ATMs while I_1^1 and I_1^2 are the accumulated interest rate for the first and second ATMs. For this problem, first, optimized solutions for two separate ATMs are calculated and used as input for calculation of grouped ATMs cost optimization.

4.3.4 Illustration of the DP Solution for Grouped ATMs on a Case

As as first step, accumulated interest cost table for the first ATM is generated as follows:

Table 4.5: Accumulated Interest Costs For The First ATM

I	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Day 1		2	4.01	13.10	17.16
Day 2			1	7.03	10.06
Day 3				3	5.01
Day 4					1

Also, accumulated interest cost table for the second ATM is generated:

Table 4.6: Accumulated Interest Costs For The Second ATM

I	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Day 1		2	8.03	20.15	24.21
Day 2			3	11.04	14.06
Day 3				4	6.01
Day 4					1

After calculating accumulated interest costs, optimized cost for both first ATM and second ATM are calculated by using accumulated interest cost matrices of those two ATMs separately.

So, generated optimized cost tables for both first ATM and second ATM are shown in below tables.

The table for the first ATM is generated like the following:

Table 4.7: Optimized Cost For The First ATM

c	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Day 1	5	7	9.1	14.1	15.1
Day 2		5	6	11	12
Day 3			5	8	10.01
Day 4				5	6
Day 5					5

In the above table, the optimized cost from Day 1 to Day 5 found as 15.1 for the first ATM.

Then, the optimized cost table for the second ATM is generated:

Table 4.8: Optimized Cost For The Second ATM

c	Day 1	Day 2	Day 3	Day 4	Day 5
Day 1	5	7	12	16	18
Day 2		5	8	13	14
Day 3			5	9	11
Day 4				5	6
Day 5					5

The optimized cost from Day 1 to Day 5 is calculated as 18 for the second ATM as seen from the above table.

As a last step, optimized cost for both ATMs is calculated by using accumulated interest cost matrices of those two ATMs and also, optimized cost matrices of them.

And, the optimized cost table for both ATMs is found as following:

Table 4.9: Optimized Cost For Both ATMs

c	Day 1	Day 2	Day 3	Day 4	Day 5
Day 1	8	12	20	27	30
Day 2		8	12	20	22
Day 3			8	15	18
Day 4				8	10
Day 5					8

And with the help of those tables, optimized costs and paths for ATM-1, ATM-2 and both ATMs are found as follows:

Table 4.10: Cash replenishment costs for all ATMs

Method	**First ATM**	**Second ATM**	**Both ATMs**
Proposed path	(13)(45)	(12)(33)(45)	<11><23><45>
Proposed method	15.1	18.0	30.0
Daily replenishment	25.0	25.0	40.0
Weekly replenishment	22.16	29.21	49.37

Here both solutions find 30 as optimum cost but slightly different paths. As seen from the table 4.9, c[1][5]=30 which means that optimum cost is found as 30 between the days 1-5 in dynamic programming and table 4.10 shows that 30.0 is found as the optimum cost by proposed method in linear programming. Please notice that, cost calculations of both of these two optimum paths are the same which is 30.

In the path definition, numbers between parenthesis show the beginning and ending day of replenishment period. For instance, (13)(45) means that cash replenishment should be done between the days 1 and 3, then between 4 and 5. This notation is used for single ATMs. For grouped ATMs, <11><23><45> in the table means that replenishment should be done together between days 1-1, 2-3 and 4-5.

Path notation for single ATMs is shown as ’(xy)’ which means that replenishment should be done between days x and y. For grouped ATMs, ’(xy)’ means that replenishment should be done between days x and y for the first ATM, ’[zt]’ means that should be done between z and t for the second ATM, and ’<pr>’ means that it should be done together between the days p and r.

CHAPTER 5

RESULTS AND DISCUSSIONS

In this section, we present the optimized cash replenishment costs by the proposed method on eight real ATMs. There are maximum, minimum and average values of proposed method, daily replenishment and weekly replenishment in the tables. First table shows the calculations made with the ATM 1-2-3-4's real data for single ATMs. Other table shows the experiments with ATM 1-5, ATM 2-6, ATM 3-7 and ATM 4-8's real data for grouped ATMs.

As the cash withdrawal prediction, we use the real withdrawal amounts. The data set contains the withdrawal amounts for about one year. Real data set of the ATMs for one year is represented in the appendix.

We construct weekly cash replenishment plans and report the average weekly costs together with minimum and maximum weekly costs obtained. Weekly average cash replenishment costs for the baselines of daily replenishment and weekly replenishments are reported as well. All the results for single ATM experiments are presented in Tables 5.1 and grouped ATMs are in Tables 5.2, where costs are calculated under the interest rate of 0.01 and loading cost of 50 for separate loading and 80 for shared loading.

Table 5.1: Single cash replenishment costs for ATMs 1-4

	Method	Avg cost	Min cost	Max cost
ATM1	Proposed method	234.37	157.0	340.0
	Daily replenishment	350.0	350.0	350.0
	Weekly replenishment	697.88	199.3	2131.35
ATM2	Proposed method	259.15	50.0	312.5
	Daily replenishment	350.0	350.0	350.0
	Weekly replenishment	1508.47	50.0	2934.60
ATM3	Proposed method	287.08	150.0	344.8
	Daily replenishment	350.0	350.0	350.0
	Weekly replenishment	1517.52	361.5	4448.09
ATM4	Proposed method	321.14	105.5	350.0
	Daily replenishment	350.0	350.0	350.0
	Weekly replenishment	5400.04	232.8	13749.0

Above table contains the average, minimum and maximum values of the costs calculated by proposed method, daily replenishment and weekly replenishment for the real withdrawal amounts of ATM1, ATM2, ATM3 and ATM4. This table shows the results for a single ATM case.

And the below table shows the same results for the grouped ATMs case. So, this table shows the results obtained from the real withdrawal amount data of ATM1, ATM2, ATM3, ATM4, ATM5, ATM6, ATM7 and ATM8. While calculating the results shown in this table, ATM1 - ATM5, ATM2 - ATM6, ATM3 - ATM7 and ATM4 - ATM8 are considered to be in the same group of ATMs.

As seen in this table, daily replenishment is different from the calculated value of the first table. The reason behind this is that the transportation cost is different for shared and separate loading. As stated before, transportation cost is 50 for separate loading while it is 80 for shared loading. Which is why the daily replenishment has different values in those two tables.

Table 5.2: Grouped cash replenishment costs for ATMs 1-8

	Method	Avg cost	Min cost	Max cost
ATM1-5	Proposed method	290.87	272.9	291.9
	Daily replenishment	560.0	560.0	560.0
	Weekly replenishment	2904.22	1221.3	9509.1
ATM2-6	Proposed method	151.8	80.0	153.3
	Daily replenishment	560.0	560.0	560.0
	Weekly replenishment	6025.43	80.0	16997.6
ATM3-7	Proposed method	281.49	233.60	288.7
	Daily replenishment	560.0	560.0	560.0
	Weekly replenishment	3065.18	512.2	8017.0
ATM4-8	Proposed method	404.21	240.0	407.1
	Daily replenishment	560.0	560.0	560.0
	Weekly replenishment	9014.39	1080.76	18205.32

It is seen from the tables that the proposed method generates schedules with much lower costs especially with respect to weekly replenishment. Daily replenishment schedule generates the same cost due to transportation on each day. This means that interest rate has no effect on daily replenishment cost at all. On the other hand, weekly replenishment cost varies depending on the amount of money required for the whole week which means that interest rate might be very important for the weekly replenishment depending on the amount required.

It is understood that weekly replenishment may not be good approach to apply for considerable amounts of withdrawal money.

Also, we have done another analysis about the execution times of the methods that we have discussed in this thesis. In this analysis, we compare the solution generation time for DP and ILP based solutions for scheduled of 5, 10, 20 and 30 days. As expected DP performs much better than ILP solution even for small problem instances. The results are shown in Table 5.3.

Table 5.3: Solution generation time comparison for DP and ILP

Days	DP(ms)	ILP(ms)
5	8	59000
10	13	69000
20	21	83000
30	50	90000

Another analysis that we conducted is on the amount of the difference between the optimized cost by the proposed method and the costs by the baseline approaches under varying interest rate.

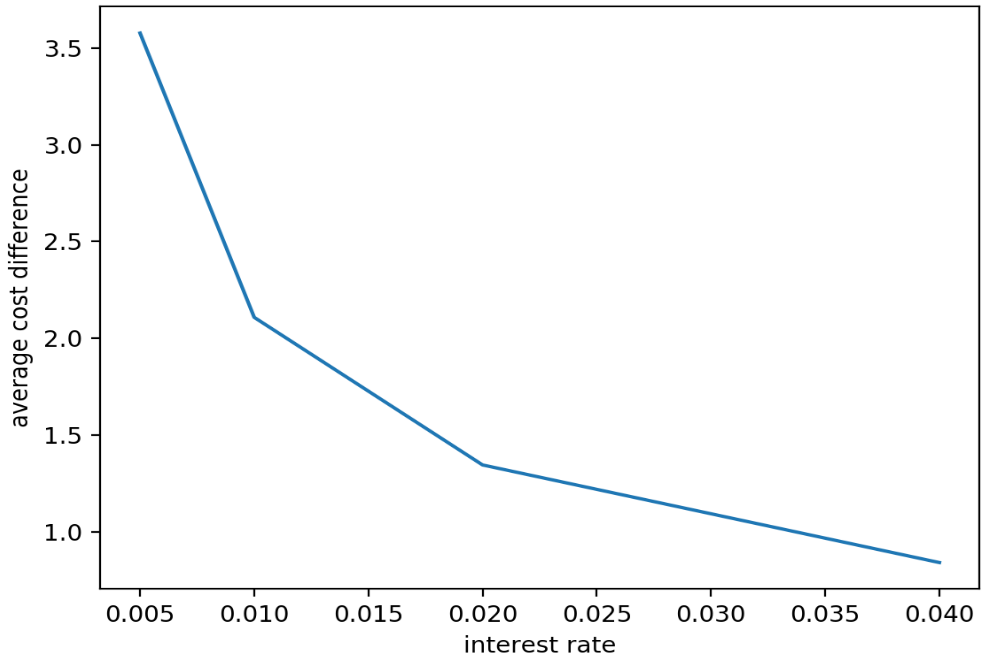
First figure shows how the difference is changing both between weekly replenishment and proposed method, also between daily replenishment and proposed method under varying interest rate.

Second figure shows the difference under varying loading cost.

As expected, the difference increases for weekly replenishment as the interest rate increases, whereas it approaches to 0 for daily replenishment. We can expect a contrary behavior for varying loading cost.

From this result, we can understand that interest rate has a huge effect on weekly replenishment while it is loading cost that affects the cost for the daily replenishment.

[Daily Replenishment]



[Weekly Replenishment]

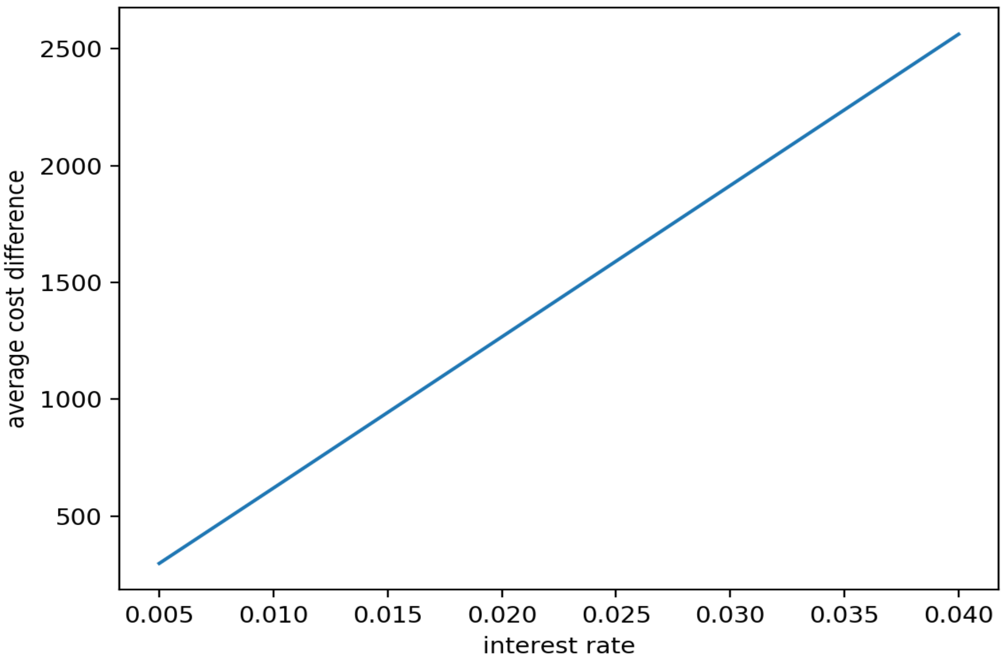
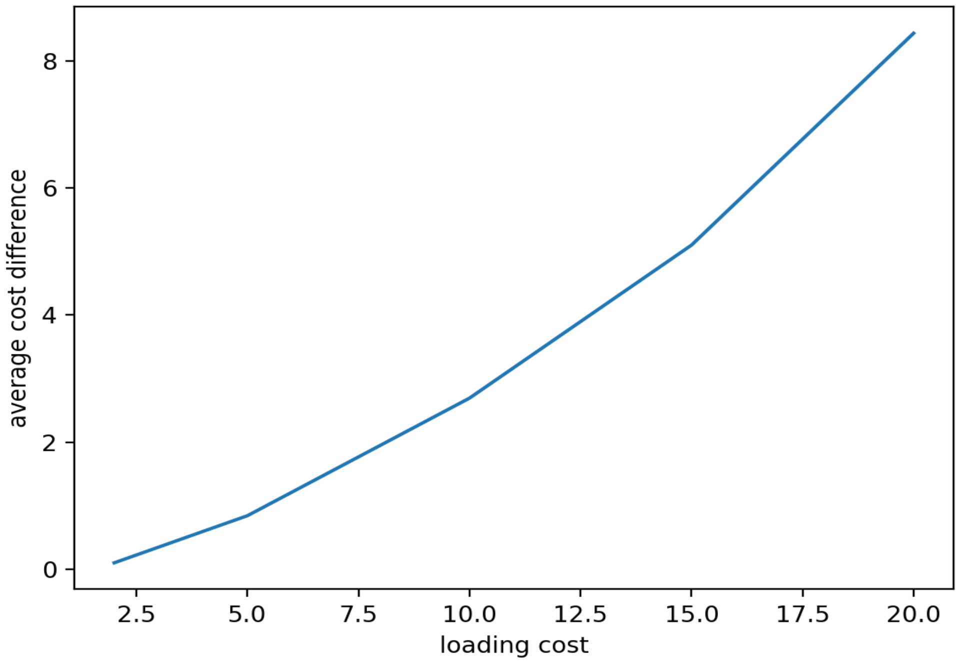


Figure 5.1: Cash Replenishment Cost Difference vs. Interest Rate

[Daily Replenishment]



[Weekly Replenishment]

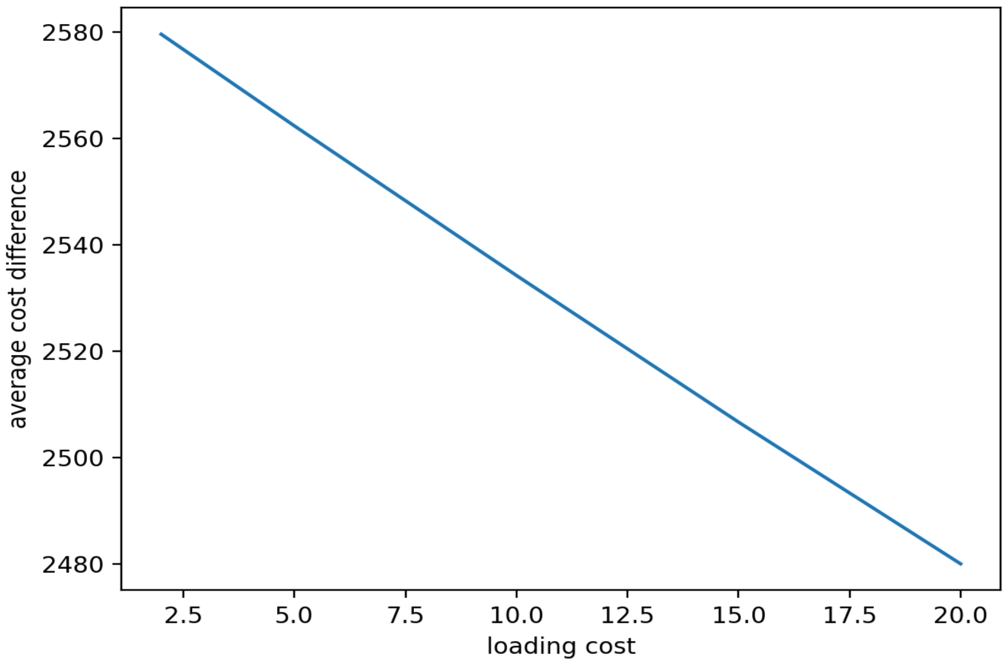


Figure 5.2: Cash Replenishment Cost Difference vs. Loading Cost

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter, firstly, dynamic programming modeling for both single ATM and grouped ATMs, and integer linear programming modeling for single ATM and grouped ATMs are concluded. After that, possible future developments which can be applied to both parts are explained.

6.1 Conclusion

In this work, we work on ATM cash replenishment problem such that, given the amount of expected cash withdrawal for a given number of days, loading cost and interest cost, the aim is to generate cash replenishment schedule such that the ATM is never out-of-cash and the replenishment cost is optimized. Note that there is a trade-off between transportation cost and idle cash cost and the solution has to find a balance between these two cost factors. We assume that the withdrawal amount predictions are available, and focus on scheduling the cash replenishment in order to optimize the cost. We present ILP and DP based solutions to the problem in a comparative way. The DP based solution is initially proposed in (Ozer, 2018). In this work, we introduced single ATM and grouped ATMs problems for cash replenishment problem and compare those with LP based solution more thoroughly. The proposed DP approach is inspired from well-known matrix chain multiplication solution through defining a mapping between matrices to be multiplied and daily ATM cash requirements. Also notice that, grouped ATMs consist of two ATMs and, the begining and the last days of the replenishment schedule is considered to be same for grouped ATMs.

In the experiments, we present optimal schedules generated on a set of real world

ATM data in comparison to baselines of daily and weekly replenishments. The results reveal that the straightforward strategies fail to find optimized cost, especially weekly replenishment generates schedules with high costs. We also present the relationship between cost difference, interest rate and loading cost. In the graphs, how the weekly and daily replenishment costs change when interest rate and loading cost change are visualized respectively.

The experiments conducted for cash replenishment scheduling for 5, 10, 15 and 30 days show that DP based approach can generate the optimized solution in about 1000 times faster than LP approach.

6.2 Future Work

In this work, cash optimization is made on real data since prediction is considered to be done already. In order to be used, prediction part can be done as well in the future.

Problem is divided into two as single ATM and grouped ATMs, but grouped ATMs are modeled with two ATMs. This can be extended with more than two ATMs. Moreover, two ATMs are considered to be close to each other without location information in this work. In the future, grouped ATMs can be decided with the location information.

Lastly, the starting and ending day of replenishment period is considered to be same for grouped ATM model in this work. In the future, new model can be established without this restriction.

REFERENCES

[1] T. Baker, V. Jayaraman, and N. Ashley, "A datadriven inventory control policy for cash logistics operations: An exploratory case study application at a financial institution," *Decision Sciences*, vol. 44, 02 2013.

[2] C.-S. Ou, S.-Y. Hung, D. C. Yen, and F.-C. Liu, "Impact of atm intensity on cost efficiency: An empirical evaluation in taiwan," *Information Management*, vol. 46, no. 8, pp. 442 – 447, 2009.

[3] A. Bar-Ilan, D. Perry, and W. Stadje, "A generalized impulse control model of cash management," *Journal of Economic Dynamics and Control*, vol. 28, pp. 1013–1033, 03 2004.

[4] J. Castro, "A stochastic programming approach to cash management in banking," *European Journal of Operational Research*, vol. 192, pp. 963–974, 2009.

[5] E. J. Elton and M. J. Gruber, "On the cash balance problem," *Journal of the Operational Research Society*, vol. 25, no. 4, pp. 553–572, 1974.

[6] R. Simutis, D. Dilijonas, L. Bastina, J. Friman, and P. Drobinov, "Optimization of cash management for atm network," *Information Technology and Control*, vol. 36, 12 2010.

[7] J.-S. Yao, M.-s. Chen, and H.-F. Lu, "A fuzzy stochastic single-period model for cash management," *European Journal of Operational Research*, vol. 170, pp. 72–90, 02 2006.

[8] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte, "Static pickup and delivery problems: A classification scheme and survey," *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, vol. 15, pp. 1–31, 02 2007.

[9] A. Kolos, G. Benedek, and Z. Gilányi, "Pareto improvement and joint cash man-

agement optimisation for banks and cash-in-transit firms," *European Journal of Operational Research*, vol. 254, 05 2016.

[10] Y. Ekinci, J.-C. Lu, and E. Duman, "Optimization of atm cash replenishment with group-demand forecast," *Expert Systems with Applications*, vol. 42, 05 2015.

[11] R. Andrawis, A. Atiya, and H. El-Shishiny, "Forecast combinations of computational intelligence and linear models for the nn5 time series forecasting competition," *International Journal of Forecasting*, vol. 27, pp. 672–688, 07 2011.

[12] M. Kalchschmidt, R. Verganti, and G. Zotteri, "Forecasting demand from heterogeneous customers," *International Journal of Operations Production Management*, vol. 26, pp. 619–638, 06 2006.

[13] V. Kamini, R. Vadlamani, A. Prinzie, and D. Van den Poel, "Cash demand forecasting in atms by clustering and neural networks," *European Journal of Operational Research*, vol. 232, pp. 383–392, 01 2014.

[14] W. J. Baumol, "The transactions demand for cash: An inventory theoretic approach," *The Quarterly Journal of Economics*, vol. 66, no. 4, pp. 545–556, 1952.

[15] M. H. Miller and D. Orr, "A model of the demand for money by firms," *The Quarterly Journal of Economics*, vol. 80, no. 3, pp. 413–435, 1966.

[16] M. Moraes and M. Nagano, "Cash balance management: A comparison between genetic algorithms and particle swarm optimization," *Acta Scientiarum. Technology*, vol. 34, 10 2012.

[17] G. D. Eppen and E. F. Fama, "Solutions for cash-balance and simple dynamic-portfolio problems," *The Journal of Business*, vol. 41, no. 1, pp. 94–112, 1968.

[18] A. Bar-Ilan and D. Perry, "A generalized impulse control model of cash management," 10 2002.

[19] J. García Cabello (JG Cabello, "Cash efficiency for bank branches," *SpringerPlus*, vol. 2, 07 2013.

[20] D. P. Heyman, "A model for cash balance management," *Management Science*, vol. 19, pp. 1407–1413, 08 1973.

[21] E. Bjørndal, H. Hamers, and M. Koster, “Cost allocation in a bank atm network,” *Mathematical Methods of Operational Research*, vol. 59, pp. 405–418, 07 2004.

[22] R. van Anholt, L. Coelho, G. Laporte, and I. Vis, “An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines,” *Transportation Science*, vol. 50, pp. 1077–1091, 8 2016.

[23] S. Chotayakul, P. Charnsethikul, J. Pichitlamken, and J. Kobza, “An optimization-based heuristic for a capacitated lot-sizing model in an automated teller machines network,” *Journal of Mathematics and Statistics*, vol. 9, pp. 283–288, 10 2013.

[24] . Bat and D. Gözüpek, “Joint optimization of cash management and routing for new-generation automated teller machine networks,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–15, 2018.

[25] P. Kurdel and J. Sebestyenova, “Modeling and optimization of atm cash replenishment,” 01 2012.

[26] D. M. D. Toit and D. Johannes, “Atm cash management for a south african retail bank,” 2011.

[27] S. Opasanon and P. Lertsanti, “Impact analysis of logistics facility relocation using the analytic hierarchy process (ahp),” *International Transactions in Operational Research*, vol. 20, pp. 325–339, 05 2013.

[28] L. Coelho, H. Larrain, and a. cataldo, “Managing the inventory and distribution of cash in automated teller machine using a variable mip neighborhood search algorithm,” 01 2016.

[29] M.-C. Chen, C.-L. Huang, K.-Y. Chen, and H.-P. Wu, “Aggregation of orders in distribution centers using data mining,” *Expert Syst. Appl.*, vol. 28, pp. 453–460, Apr. 2005.

[30] R. H. BALLOU, “Measuring transport costing error in customer aggregation for facility location,” *Transportation Journal*, vol. 33, no. 3, pp. 49–59, 1994.

[31] A. Zarnani, M. Rahgozar, C. Lucas, and F. Taghiyareh, “Effective spatial clustering methods for optimal facility establishment,” *Intell. Data Anal.*, vol. 13, pp. 61–84, 01 2009.

[32] C. F. Daganzo, “The distance traveled to visit n points with a maximum of c stops per vehicle: An analytic model and an application,” *Transportation Science*, vol. 18, pp. 331–350, 11 1984.

[33] V. Gaur and M. L. Fisher, “A periodic inventory routing problem at a supermarket chain,” *Operations Research*, vol. 52, pp. 813–822, 12 2004.

APPENDIX A

REAL ATM REPLENISHMENT DATA USED FOR EXPERIMENTS

Contains tables of eight ATMs' real withdrawal data.

A.1 Withdrawal data of ATMs 1-8

Table A.1: First ATM real data.

Weeks	**Day-1**	**Day-2**	**Day-3**	**Day-4**	**Day-5**	**Day-6**	**Day-7**
1	1230	50	1580	30570	11370	4450	1180
2	1700	2780	4350	1970	3710	4100	1190
3	130	3210	80	2920	1800	3560	970
4	600	3440	2460	2190	3200	3640	3340
5	630	1315	1570	2720	2930	29620	5950
6	2740	6610	3140	1030	1540	2650	140
7	100	8870	6220	5170	690	2770	2430
8	20	6650	1580	3250	1930	1050	640
9	200	4220	6620	8680	4780	6890	6720
10	400	13510	7980	8290	2430	5030	5200
11	30	1140	3770	4620	4430	2960	3940
12	290	6710	5720	740	1700	3280	1730
13	100	510	1810	480	2440	1360	820
14	160	300	20270	14730	5760	3270	1440
15	80	1600	1570	580	960	30	2770

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
16	50	1630	350	490	6300	14060	650
17	14420	2610	4210	4750	4440	1300	440
18	2390	1050	4480	370	5810	790	3300
19	230	2310	2640	1090	2240	1220	550
20	490	250	20	380	4860	1920	3570
21	4220	2030	1200	320	2160	3060	5160
22	3430	2070	100	1350	40	730	800
23	5470	710	100	700	1930	2540	2790
24	9510	7370	20	2020	1290	2140	3740
25	30	2890	80	6810	3360	3620	4620
26	100	1850	700	340	1300	3280	1890
27	640	3250	2720	1310	980	2730	1700
28	7790	7000	6160	7410	8720	2390	1770
29	900	1500	2190	1340	1500	8570	640
30	1050	2660	3490	1010	830	380	550
31	2980	430	4380	1210	220	540	500
32	23530	13500	5370	3920	1100	2500	2440
33	990	2040	3120	670	3380	620	3500
34	1680	3090	1040	380	2940	1100	550
35	3920	1780	430	590	5050	1530	4250
36	10750	1300	16220	90	4200	790	3260
37	720	720	5370	3230	6130	5690	1760
38	210	3710	1290	290	1250	4260	3380
39	400	8150	3190	1550	3990	720	2470
40	11170	2280	3790	1620	100	3830	20
41	710	6130	7150	2160	110	2700	1680
42	1990	1920	1110	460	3280	50	940

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
43	2130	4970	2520	290	480	2190	27620
44	11970	1820	50	1240	1300	3150	5980
45	780	170	3120	3330	3630	2480	2120
46	850	2460	50	1000	1830	900	870
47	410	2080	1250	3420	1320	2020	3410
48	2970	10370	7000	2400	730	5340	330

Table A.2: Second ATM real data.

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
1	2450	1250	9790	13940	8090	8570	9790
2	1000	100	9880	3290	3330	5620	10640
3	50	200	10880	10090	5450	6620	6650
4	2200	100	12955	7550	9700	7900	6000
5	4180	19210	4730	22320	500	16670	9350
6	2950	3290	8040	350	7620	3190	4850
7	990	8650	260	200	2590	4850	1530
8	6420	7870	150	240	12300	19130	8850
9	11000	6770	110	20	17680	6260	8640
10	10340	7420	1140	320	6230	10170	2760
11	8970	10640	330	11070	4350	7340	10520
12	12460	130	13250	12040	14600	10120	18770
13	670	650	16660	7710	10710	16020	18070
14	9180	7730	3590	19510	480	10000	11350

Continued on next page

Weeks	**Day-1**	**Day-2**	**Day-3**	**Day-4**	**Day-5**	**Day-6**	**Day-7**
15	11880	9100	8980	100	14180	9650	3400
16	4510	7340	730	20	3390	10680	9500
17	11640	14400	1730	80	16580	18050	5300
18	5810	13500	150	5870	7710	5625	220
19	19360	8750	10860	500	23005	8880	5620
20	5140	130	7410	680	17070	9540	10280
21	80	12250	180	3370	8870	8650	380
22	16170	400	7020	16130	33715	1400	3850
23	9130	710	8390	14670	20380	50	10150
24	430	11270	8670	14100	800	200	5310
25	1250	13610	6110	23000	2870	880	5190
26	1810	14530	10960	12090	150	1630	15310
27	820	11220	6000	11110	8110	120	9740
28	11850	20140	600	5580	80	3620	7650
29	12170	290	80	6020	13530	11530	8870
30	810	7760	800	11700	16700	18400	110
31	300	4190	9350	1870	5220	9020	7820
32	620	20	5100	20	4100	10400	11090
33	640	200	6640	1020	11660	9955	10180
34	70	150	14230	3500	5300	6400	340
35	40	9340	370	7010	10140	6150	150
36	12150	40	6410	7760	6020	160	11335
37	200	27690	11640	4950	1080	490	16170
38	640	11250	4000	12120	200	20	6310
39	600	6480	3560	8780	150	5300	400
40	11550	6440	6430	140	150	6120	540
41	15550	20110	11110	1270	590	9030	20

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
42	9000	5900	6900	900	5965	500	9480
43	4660	8570	160	30	5340	60	11350
44	9990	21090	250	100	9470	70	11885
45	10120	19140	130	190	9700	3370	4470
46	8915	15940	90	8170	100	8900	8460
47	12390	440	5960	450	7865	14850	10630
48	400	9910	60	4630	5840	8170	20
49	5900	460	13450	26200	13620	500	50
50	10160	200	7530	7420	16090	900	8470

Table A.3: Third ATM real data.

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
1	7140	4480	8490	11470	15070	8850	9730
2	19600	13740	11340	4190	3090	2060	2060
3	7670	7810	5310	4980	5720	6750	140
4	3520	6140	12290	15020	14580	630	4680
5	3260	1280	15560	16630	550	6070	5500
6	5200	220	4090	3650	6270	11190	8710
7	1600	5400	6760	9070	17180	15680	3300
8	14380	13120	11920	6450	8460	3000	6850
9	6750	5280	6930	13860	600	9430	8870
10	4090	10940	2940	1260	4150	3750	6020
11	200	11880	14920	10140	12140	1500	18890

Continued on next page

Weeks	**Day-1**	**Day-2**	**Day-3**	**Day-4**	**Day-5**	**Day-6**	**Day-7**
12	9510	9050	6710	7840	3410	4510	3510
13	2720	7770	5230	2710	4655	2280	3110
14	250	1830	19170	18530	11750	10010	18090
15	10610	5700	7430	540	3580	8000	6280
16	6330	6550	2700	3090	5670	6620	5240
17	20	3400	8850	12960	10830	14550	14510
18	10760	6480	5260	5100	4190	4280	1530
19	9110	3230	26060	4630	50	6300	11700
20	19560	90	2090	16570	1500	21310	7300
21	4550	4360	4240	7340	6400	3090	12880
22	4110	7360	290	5560	350	16820	8780
23	17030	400	7650	1500	12270	5900	10810
24	160	9390	4730	8610	12960	1800	1500
25	3360	2830	5580	1680	7080	4820	10490
26	17800	19270	10810	111660	8510	3820	1800
27	4100	300	4010	4380	2320	6490	1400
28	9220	400	15320	21190	8070	10470	1500
29	8490	14030	300	12780	3260	3570	4960
30	2030	50	1690	3590	4160	13110	15040
31	30720	400	15720	1500	8530	7850	6870
32	2760	6000	2830	5770	250	3010	5660
33	2620	10070	400	3190	50	4930	5280
34	7910	11820	20	11090	9220	6210	120
35	3550	4020	4870	2710	1320	20	4350
36	2720	5690	6160	50	2450	4220	3260
37	8790	6150	20980	24000	10330	10	9960
38	11080	9220	7810	250	7590	3490	7410

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
39	4850	20	4570	2390	8440	6600	20
40	4160	4500	7270	590	14000	1500	10950
41	9730	13070	9040				

Table A.4: Forth ATM real data.

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
1	31070	81090	18360	88330	57100	34910	29770
2	37350	17530	4890	46570	25230	17530	18760
3	34180	21750	2520	21440	23860	23640	13460
4	32540	15180	3640	19030	8310	11040	14370
5	19480	15220	8240	28190	116240	40120	47090
6	27270	28630	15460	29720	9240	16450	33400
7	29740	23150	5980	41960	33940	20220	5190
8	25350	17610	12840	7790	14970	16470	17560
9	23620	17960	4560	22520	100450	95630	65885
10	70190	21160	12330	39590	29300	29140	33190
11	25320	19460	3170	13110	19530	22370	28760
12	33370	6360	6230	28500	19800	16710	7610
13	20190	14810	3740	14230	2000	23790	24690
14	95000	102380	31400	8000	57230	36420	22580
15	29010	14230	5040	46900	21740	29320	23080
16	28670	18830	3650	21900	15350	17590	19920
17	21560	12430	5700	18190	22410	15470	24160

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
18	20340	18570	3540	1410	146520	54930	39100
19	56490	26260	7270	25080	36210	32310	21600
20	24070	11990	4000	20750	25510	17220	11470
21	22390	15230	8040	25700	17270	22010	16340
22	13420	12470	3500	47130	75515	67350	37280
23	10460	26620	1000	21775	17590	32890	18990
24	2500	12810	310	11550	17610	8770	3230
25	2110	15650	190	6720	23890	32090	18490
26	4790	26150	390	20490	22750	63860	128300
27	31130	62590	800	10870	7520	31370	22900
28	4860	48050	3440	21420	29540	40320	50550
29	2620	26930	730	21010	28970	17370	22310
30	5710	19480	850	20200	17300	25100	13250
31	5600	81550	12890	62125	28820	48650	33110
32	14375	25650	240	33250	20405	27160	15890
33	8190	32990	1330	33540	26410	15360	20060
34	6190	24600	780	19260	29670	20210	10020
35	3060	13040	560	11190	80080	104250	64490
36	22800	30910	45350	38680	68990	33890	13060
37	14780	290	7130	6630	20510	16670	5390
38	26260	2100	12890	19320	24390	15960	2520
39	22500	960	25300	14340	30390	90100	32210
40	66220	21520	49190	6430	530	22860	2230
41	21750	21400	22020	18360	6610	27190	1730
42	18200	22740	36210	20050	3670	24450	2730
43	21520	14500	23380	13640	9890	19830	1850
44	121070	76610	45360	23210	8800	40900	2830

Continued on next page

Weeks	**Day-1**	**Day-2**	**Day-3**	**Day-4**	**Day-5**	**Day-6**	**Day-7**
45	29340	41800	26490	23590	5290	22820	410
46	18400	28180	31640	26630	8530	25880	310
47	15610	18980	26770	17340	2750	11310	2230
48	19180	99070	143690	44120	19580	58050	2690
49	30020	31500	21850	23120	5100	22600	350
50	24220	33460	27720	18470	5940	30290	330
51	18940	16720	19080	19470	5870	23770	400
52	12060	22630	24660	14040	3890	148570	16000
53	59360	37940	26490	14440	30210	23520	31930
54	16180	5940	22800	1380	23070	18485	24930
55	22460	3450	28050	4160	20050	35530	15890
56	13290	10770	13390	108530	104790	76280	29680
57	5320	52120	2420	21500	24040	24010	23550
58	2600	17180	550				

Table A.5: Fifth ATM real data.

Weeks	**Day-1**	**Day-2**	**Day-3**	**Day-4**	**Day-5**	**Day-6**	**Day-7**
1	100	50	90	5290	6430	81820	53710
2	19190	4660	60	14700	10190	12340	9400
3	14010	3550	180	8420	4250	13860	6680
4	11930	1130	50	7130	5760	3200	7140
5	4360	150	9050	3780	9190	2020	73290
6	26010	1230	39110	8010	6370	16150	11290

Continued on next page

Weeks	**Day-1**	**Day-2**	**Day-3**	**Day-4**	**Day-5**	**Day-6**	**Day-7**
7	5610	240	9720	4550	5430	2540	12350
8	1940	3040	4370	8380	4200	6030	1490
9	50	7890	3330	6790	4330	104090	10350
10	1710	42390	21100	10960	12870	12610	3660
11	50	11400	9970	9680	6720	10090	1370
12	90	11130	4350	9080	7220	6940	1040
13	570	10010	9420	8020	10630	10800	140
14	5850	88960	57270	42830	3760	19260	14440
15	6500	11810	11680	290	140	8080	8920
16	9350	5320	9990	980	1000	18590	7290
17	3790	7010	8450	2010	70	1220	7590
18	5370	4870	100270	20530	3660	34320	20900
19	11780	6540	15240	770	110	8060	6550
20	5960	160	11990	780	130	11470	5420
21	7650	7280	4760	630	50	6150	700
22	4700	5820	5960	1300	100	54650	13990
23	45690	23880	19140	1670	150	13140	530
24	8380	7170	15170	1340	150	7720	90
25	3120	7270	15580	1320	20	12590	160
26	4760	6930	48180	7490	2070	19050	1000
27	56970	980	44900	1230	7150	11300	12970
28	1980	110	8340	610	6100	3750	7550
29	170	9890	7060	6250	12500	800	10840
30	4150	3960	73090	4660	2230	43380	2040
31	16090	9160	11330	620	970	13480	50
32	8120	9970	7450	2240	11090	4660	10740
33	2420	40	6480	7350	6700	12360	600

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
34	100	7290	40	60610	39890	29720	2660
35	700	100	750	100	10650	90	12055
36	5640	5760	12980	1410	30	8740	40
37	7260	5330	9520	870	50	10130	540
38	3925	104180	72400	10200	280	8930	7790
39	10830	7050	1740	100	10420	300	8270
40	6070	12110	1130	420	14370	1310	4960
41	6980	11730	260	190	3000	1100	10040
42	1610	24280	14250	1650	62990	3300	18170
43	14610	17830	2110	550	9990	370	9910
44	9335	11450	1210	1240	9490	1060	6830
45	5540	12350	3300	50	6700	150	8850
46	2910	10970	1040	210	4560	210	57090
47	34080	26560	4140	40	10050	540	5810
48	11270	13720	200	190	13600	100	7480
49	5560	11220	590	11480	1700	8170	6580
50	6360	990	3270	2370	106965	9200	1090
51	51360	120	17470	17040	14300	1050	50
52	17230	260	5980	9740	4450	1770	170
53	8100	13280	2730	12670	580	40	9980
54	150	9050	3740	3850	140	490	113350
55	4410	23450	11820	15780	2570	40	15890
56	270						

Table A.6: Sixth ATM real data.

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
1	3040	2330	8380	15030	192290	48760	69670
2	34400	16580	1780	28090	15210	18840	17760
3	25290	10780	3650	22010	22840	23440	18800
4	22830	6850	3780	10560	12150	16910	3120
5	7880	19810	9710	10520	13960	168530	25790
6	22030	65070	14320	5680	21630	36990	6890
7	2970	23900	26560	20690	1630	15790	6220
8	1930	8260	19640	12940	15000	23215	5030
9	7290	22500	8980	18510	8890	15020	28550
10	19150	123420	69310	49950	28740	31030	19210
11	5940	29010	21000	25500	15990	26460	4860
12	2560	13400	18880	13140	13870	18240	5120
13	2130	22080	8670	12350	8690	19940	5290
14	3820	640	173520	44430	25110	17170	11960
15	42520	19370	10200	19320	3980	3440	4100
16	18710	26650	24610	14670	22340	6870	1030
17	19860	15240	13250	19260	24830	2640	1430
18	90	12630	13520	211000	75960	21690	6570
19	42890	35610	24910	20360	32820	8790	3120
20	34830	17580	20420	6110	29780	2500	3540
21	21370	19910	16120	8320	20660	2750	2780
22	16230	4170	15390	11910	22200	3560	4610
23	167670	9090	48320	34990	30660	2690	4760
24	28820	24550	23450	36330	12600	2520	31580
25	2020	20630	23020	19910	4720	2430	14230
26	60	16660	19710	34290	10080	4050	127880
27	2270	27030	10610	40790	9350	15630	51710

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
28	1800	31110	19360	42140	12620	6010	30100
29	40	26340	23510	21260	3970	3030	14190
30	140	15900	15550	18940	4310	1880	10140
31	13660	11390	167880	15990	7830	36280	1750
32	45360	22070	29870	10110	5970	27940	1530
33	13760	14300	26510	5320	1870	15040	1440
34	14160	10730	25970	3530	6130	8690	7990
35	15390	18460	3880	3760	134470	57150	32840
36	65970	25480	8850	9430	50	3820	7070
37	19040	10970	3260	27500	23020	20280	23430
38	3890	3980	21270	2100	12560	11360	15530
39	8670	1780	22500	20	186340	2110	53670
40	9760	13880	32800	470	21120	19070	28860
41	10110	6800	32750	150	21150	22930	24830
42	5680	3290	23760	1100	8190	8830	14170
43	7530	1630	21110	120	14770	22440	16660
44	4770	2595	195690	11270	46380	34940	20220
45	5430	8810	34230	2010	20290	19630	25460
46	8170	4230	24250	2430	19500	20250	33950
47	7830	2490	19660	940	17060	12440	12505
48	7130	1300	150220	53520	29340	41370	14640
49	3480	28250	220	29420	20020	37590	8200
50	5820	22420	2510	15720	16610	21040	6360
51	1790	27970	1160	17890	15940	18810	6110
52	550	10500	430	23680	187990	50850	14620
53	18720	48170	3570	37650	25800	22630	12530
54	4550	40100	700	11480	20440	29430	3800

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
55	6550	20600	800	16450	14400	30300	6800
56	4530	22800	3400	10020	32630	28740	5320
57	1960	193410	7340	34780	36530	48210	10240
58	6350	27400	330				

Table A.7: Seventh ATM real data.

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
1	6790	1470	71550	38720	19530	9810	1910
2	6070	6110	9050	10020	4230	1860	2790
3	7770	1830	5240	150	3770	8830	590
4	3810	4210	890	200	2030	4510	1770
5	5300	3700	10520	540	32950	13140	4900
6	9910	2430	190	7130	5150	5820	1280
7	9110	410	4850	3110	4830	7360	6290
8	100	4160	1930	400	2250	4300	300
9	100	86410	40900	14990	11950	7510	1850
10	7080	13410	7970	4000	11160	470	3870
11	6270	2730	3570	4160	150	1260	3590
12	1670	1100	1890	600	60	76600	50090
13	21000	6840	2470	1500	21830	8470	6950
14	5760	4270	2300	550	4010	4960	5240
15	2900	5820	20	200	5360	5070	4460
16	2740	1640	460	130	7310	3560	58800

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
17	52300	9070	390	15740	27570	10270	5030
18	5280	1000	200	6250	6490	6490	2640
19	5580	4460	3910	2200	4880	3380	2050
20	120	1900	4140	4060	1880	6640	300
21	75750	20010	17390	10400	9880	20390	250
22	6750	170	9330	2160	6560	3970	7390
23	220	6690	6410	6250	3540	7830	30
24	2240	7770	52140	2710	90	700	12380
25	400	26530	10280	8040	2760	840	5450
26	70	1110	4160	3810	1250	300	5830
27	1350	3450	8980	3930	1100	100	1630
28	190	2460	3700	94070	3330	590	19950
29	1280	6310	1260	21110	240	400	11920
30	860	3730	3500	3010	100	1090	6970
31	190	3200	4210	13010	150	8770	2450
32	3190	4720	760	1500	3940	19240	10380
33	28880	1870	500	200	7100	680	2560
34	4410	2010	1420	590	5280	350	1320
35	2340	5730	1750	100	8090	350	2880
36	105510	30470	3000	120	12040	300	5840
37	2870	25090	6080	200	5080	2070	9160
38	3840	6220	5380	2700	180	6210	1710
39	4280	60	250	3020	2760	3750	3590
40	3290	110380	13840	14600	5550	24170	10210
41	11120	150	5960	1580	7470	7860	7090
42	1760	980	4570	6840	800	500	4430
43	340	2280	1910	4670	2810	260	3760

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
44	1910	107910	39510	17610	3370	2710	4380
45	260	8520	12370	9590	1140	290	5350
46	320	2590	2370	3990	1550	2800	8560
47	1050	4240	4260	6780	400	590	3900
48	380	6840	7190	110210	5460	500	24150
49	1000	12220	3330	10810	3980	730	5650
50	10	5960	1620	9120	3610	2970	270
51	3590	7480	3490	1570	3000	60	1410
52	4270	2930	3400	50	72110	7200	11540
53	1870	3130	520	4580	350		

Table A.8: Eighth ATM real data.

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
1	8810	7920	3060	20550	88530	21610	19380
2	26310	22230	11480	25900	15540	8200	17230
3	16590	16810	5090	14006	16700	9470	15410
4	22060	12220	7530	8170	10200	10810	8630
5	12100	8660	8820	17130	17940	19880	11410
6	66020	52090	12330	27260	11243	11350	15220
7	16680	13650	8780	28853	10930	17250	3491
8	13780	15315	13480	4010	13890	11690	19800
9	16641	11830	9310	12300	21050	16920	19770
10	42045	22080	12550	29010	48080	51760	34365

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
11	24720	22600	19260	13010	6420	8330	10790
12	23423	21490	16940	19720	18400	13800	6390
13	11802	19432	14580	16465	19890	15880	5130
14	2830	37430	21195	25080	40990	21300	9660
15	29120	22060	34830	31860	21660	15585	7790
16	13654	21500	25740	20790	25830	8950	4260
17	13960	14710	13950	19090	14795	13080	8270
18	1950	39440	26944	17380	44200	22980	11920
19	22990	28020	26190	40380	22440	16860	6680
20	22020	20390	13200	16730	16280	15420	8000
21	11810	15810	21240	20490	25610	14810	10610
22	12073	720	27490	37700	38410	15970	6820
23	10570	44910	49230	36760	21610	10410	20720
24	4540	16710	7740	16355	14440	7690	20130
25	740	10380	14810	15090	17370	3260	12130
26	880	20060	52870	50202	28150	11430	13090
27	410	1810	3620	8050	9130	6810	19460
28	1700	65120	36900	32320	34260	8550	18140
29	150	24150	14240	19530	8750	3530	19190
30	1610	11950	16300	18270	10030	6770	19560
31	6950	8660	41410	32570	27860	7200	20390
32	260	65340	23570	23190	20540	4210	19380
33	20	23380	24510	22580	23680	10360	23903
34	240	12940	14280	29420	12070	3980	19480
35	70	13060	28910	41290	33310	13643	23348
36	2770	63890	59840	60652	19310	13980	9390
37	80	6230	8490	17450	8010	5570	16370

Continued on next page

Weeks	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7
38	610	18380	14460	17900	20610	5059	12930
39	1610	18140	13730	17450	20900	6970	13678
40	4200	38730	23130	33820	14430	11790	19380
41	260	63540	33309	44720	18630	8080	33890
42	22470	15590	20810	17870	11240	22320	1650
43	15180	13150	24000	22630	9630	27170	80
44	27290	23244	31280	18460	12900	55268	999
45	70240	50220	32570	24500	11250	27410	1420
46	24060	22330	29320	24500	9270	25950	20
47	20470	9450	20340	18820	14500	20580	390
48	15820	32950	29899	26135	7690	19070	140
49	19100	23450	19310	14440	6250	15960	1500
50	85610	36960	37620	26320	10320	21510	1790
51	19860	17800	16740	16830	10810	15440	130
52	18681	7815	19450	14670	5780	19960	470
53	47090	17990	10360	24230	11290	35640	440
54	28800	16725	25031	36470	17890	20770	530
55	13450	17140	31088	22220	10210	20740	140
56	18920	19730	18010	16050	15740	17350	70
57	31860	14090	32150	20640	14470	71186	4730
58	44042	32096	24850	13230	9150	22726	2160