Development and Performance Evaluation of Storage Allocation Models in Warehouse

By

Md. Saiful Islam

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Industrial Engineering and Management

Khulna University of Engineering & Technology
Khulna 9203, Bangladesh

November, 2018

DECLARATION

This is to certify that the thesis work entitled "Development and Performance Evaluation of Storage Allocation Models in Warehouse" has been carried out by Md. Saiful Islam in the department of Industrial engineering and management, Khulna University of engineering & technology, Khulna, Bangladesh. The above research work or any part of the work has not been submitted anywhere for the award of any degree or diploma.

[illegible] 8/11/18

Signature of Supervisor

[illegible] 08/11/18

Signature of Candidate

APPROVAL

This is to certify that the thesis work submitted by Md. Saiful Islam entitled "Development and Performance Evaluation of Storage Allocation Models in Warehouse" has been Approved by the Board of Examiners for the partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Department of Industrial Engineering and Management, Khulna University of Engineering & Technology, Khulna-9203, Bangladesh in November 2018.

ACKNOWLEDGEMENT

First of all the authors would like to convey his gratitude to the almighty Allah without his blessings not a single activity on earth come to in success.

The author would like to convey his gratitude and respect to his respected teacher and supervisor Md. Kutub Uddin, Professor, Department of Mechanical Engineering, KUET, Khulna for providing precious guidance, inspiring discussions and constant supervision throughout the time of this work. His help, guidelines, supervisions, constructive criticism, and efforts made it possible to present the work contained in this thesis. The author is also grateful to all the other teachers and staffs of the Department of Industrial Engineering and Management, KUET, Khulna for their cordial cooperation during the course.

The author also would like to thank the authority of Khulna University of Engineering & Technology to give the support.

November, 2018 The Author

ABSTRACT

Order picking is the most labor-intensive and costly operation in warehouse. Efficient storage assignment rules enormously affect the order picking efficiency. Warehouse managers are concerned about searching for the most effective way of managing order picking, minimizing the costs involved in terms of travel distance. The purpose of this research is to develop an efficient dedicated storage allocation method utilizing both the principle of group technology (GT) and the cube-per-order index (COI) storage policy in a maintenance environment warehouse. In this method, spare parts required for a particular maintenance work are grouped together on the basis of similarity of usages, even if they are dissimilar in size, shape etc. After that, the groups and parts within each group are assigned according to COI rule. The performance of the method is evaluated using simulation and compared with other sequence, frequency and group based dedicated storage methods. A genetic algorithm is embedded into the simulation program to determine the optimal traveling sequence that will minimize the travel distance. In the simulation study, it is found that the proposed method has the highest reduction in the distance travelled by the pickers.

.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Warehouse

In any supply chain facilities, warehouse is a vital portion. The primary roles of a warehouse includes buffering of material flow to accommodate variability caused by product seasonality, consolidation of products from different suppliers for joined delivery to customers, and value-added-processing such as packaging, labeling, and product customization [1]. All the problems that arises in the warehouse are categorized into two main groups: the problems related to design and the operational problem. The design problem is concerned about the design matters such as the whole structure, dimension and sizing, layout of the departments, selection of equipment, and strategy of operation of the warehouse. On the contrary, the operational problem deals with receiving, delivering, storing and picking orders.

1.2 Warehouse Operations

The main operation of a warehouse is the movement of materials into the warehouse. Goods are unloaded from transport carrier at the receiving docks. After verifying the quantities, a random quality checks are performed on the receiving loads. Subsequently, the incoming loads are transferred to the storage locations. The form of storage will depend on the size, quantity of the products stored, and the handling characteristic of products or their product carriers [2]. Whenever an order is requested, some specific items of that order must be retrieved from storage area. This process is called order picking. Sometimes, it is required to accumulate and sort when orders have been picked in batches. Accumulation and sorting of items must be performed before being transferred to the shipping area. The information transfer is another important function of warehousing. It occurs concurrently with the movement and storage operations. Warehousing information such as stock level, different SKU's locations, information of suppliers and customers etc. is very important for administering the warehouse operations. Figure 1 shows the typical operational areas and flows within warehouse.

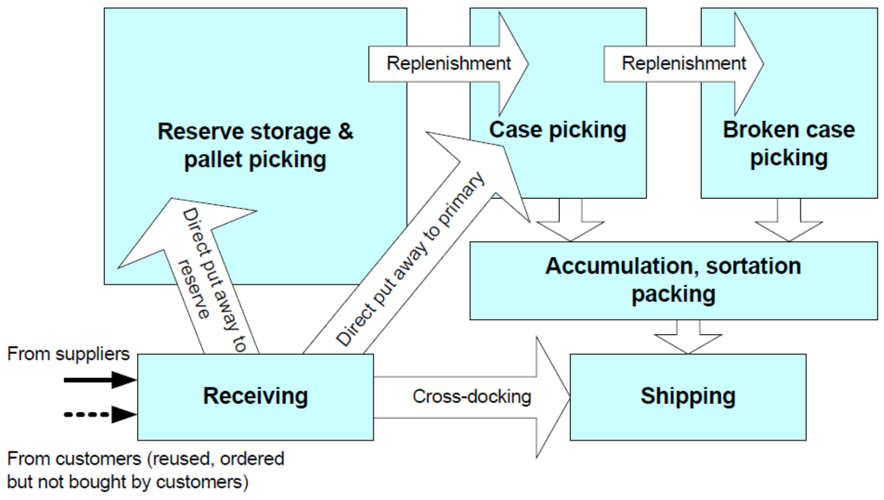

Reserve storage & pallet picking
Replenishment
Case picking
Replenishment
Broken case picking
Direct put away to reserve
Direct put away to primary
Accumulation, sortation packing
From suppliers
Receiving
Cross-docking
Shipping
From customers (reused, ordered but not bought by customers)

Figure 1: Typical warehouse operations and flows [2]

1.2.1 Receiving and shipping

The interface of a warehouse are receiving and shipping for incoming and outgoing of material. Receiving operations accept delivery from docks where materials are unloaded to storage locations within a warehouse. Shipping operations transport items from the storage locations to the mounting area where they will be loaded into the carriage vehicles, ordinarily trucks and railroad cars. Both receiving and shipping operations are influenced significantly by the equipment selected and the pattern of movement applied to shift the products, either from the receiving docks to warehouses, or vice versa [3]. For conventional warehouses which hold inventory, received goods are put away into storage and later taken and transported through shipping area. Consequently the receiving and shipping operations are much complex to control as they are dependent on the storage and order picking function. For instance, the timing of carriage vehicles may be directed by batching of orders and allocated to picking waves and vice versa. The focus of the researches on receiving and shipping is carrier-to-dock assignment problem in the matter of cross-docking warehouses, assuming statistical information regarding inward and outward shipments [1].

1.2.2 Storage

Storage is the principal function of a warehouse that concerns with the arrangement of goods kept there. The purpose of the storing function is either to capitalize resource utilization in the mean time trying to maximize satisfaction of customers concurrently [3]. Three ordinary storage policies are in practice in the warehouse namely random storage policy, dedicated storage policy and class-based storage policy [4]. The randomized storage policy defined as allocating all received products into a place that is determined arbitrarily from empty positions. The advantages of this policy are the even uses of the storing spaces and decreases pathway blocking. However, this policy consumes huge travel time because the pickers must navigate the entire warehouse. The dedicated storage policy gives a specific location to each product, i.e., this location is reserved even if the product is out of stock. Dedicated storage locations assist to surge the orientation of the order pickers, resulting in larger routing velocity and lesser incorrect picks. Dedicated storage policies permit logical classification of stored items that is frequently helpful for retail warehouses and generally for items with different weight, which can be kept in decreasing weight along the standard picking route, indicating a decent stacking order [5]. However, the drawback of this policy is the lowest space utilization amongst all storage policies. Volume-based storage assignment is a particular type of dedicated storage assignment scheme in which items are assigned to storage positions depending on their pick volume; typically high pick items are placed nearest to the input/output point (I/O point). The idea of class-based storage integrates some of the schemes stated so far. Items are congregate into groups to such a degree that the fastest moving group comprises about 15% of the products held while contributing to about 80% of the turnover. Every cited groups is assigned to a dedicated section, and products are placed randomly within their area. A-items are the fastest moving; next fastest moving class is known as B-items, and so on. As a result, class-based storage requires more space compared to random storage policy. The major benefit of this strategy is the drop of travel period [6]. Furthermore, Heskett [7] introduced a new policy called the cube-per-order index (COI) storage assignment policy dealing with the ratio of the maximum allocated storage space to the number of storage/retrieval operations per unit time. The COI policy considers both a SKU's popularity and storage space required by it. The dissimilarity between this method and COI-based storage is that the volume-based assignment only takes the popularity of items into consideration, leaving aside their space occupation. Product categories are ranked by increasing COI value and the classes with the lowest COI are stored in the most wanted locations. The COI rule is easy to apply and has the instinctive appeal of detecting compact, fast moving items in readily accessible locations. No assignment policies talked over so far have entailed possible relations between products. For illustration, clients may tend to order a specific product alongside one more product. In that instance, it is stimulating to locate these two products in nearby locations. This is an example of family-grouping, where similar products are placed in the identical region of the storing area. Evidently, grouping of products can be combined with some of the previously stated storage policies. For instance, it is possible to combine class-based storage and simultaneously group related items [6].

1.2.3 Order Picking

Order picking is an operation that refers to the retrieving of items from storage locations in response to specific orders of customers. Order picking has been practiced as the most labor-intensive and expensive operation of most warehouses from long ago. Bartholdi and Hackman [8] assessed the picking operations cost 55% of the total operating cost of a warehouse roughly. Illogical execution of order picking can cause inadequate service and high operating cost for the warehouse. For these reasons, order picking has become an intent area to warehousing experts for improving efficiency in warehouses. Different order picking system types is used in warehouses. Most of the warehouses hire humans for order picking purpose. The most common picking system is the picker-to-parts system where the order pickers walks or drives along aisles to pick items. Automated storage and retrieval systems (AS/RS) are included in the parts-to-picker system, using mostly aisle-bound cranes that retrieve one or more unit loads and bring the loads to a pick position. An AS/RS system is a storing system that make use of fixed-path storage and retrieval equipment running on one or more rails between fixed arrays of storage racks. AS/RSs are used to store and retrieve items in different settings. Racks, cranes, aisles, I/O-points, and pick positions are the main constituents of an AS/RS. Racks are metal constructions with locations that can house loads that need to be stored. Aisles are created by the empty places between the racks, where the cranes can pass through. An input/output point (I/O-point) is a specific place where retrieved loads are unloaded, and where arriving loads are received for storing. Pick positions are places where individual items from a retrieved load is taken out by people who are working before the load is sent back into the system. There exist large amounts of system options for AS/RSs. An AS/RS has in each aisle one crane, which cannot abandon its own aisle (aisle-captive) and which can carry only one unit of load at once in the simplest version. Here product handling is only by unit-load; people are not engaged in handling distinct products. The racks in its simple form are static and single-deep, that means every item can be accessed directly by the crane [9].

1.3 Genetic Algorithm

Genetic Algorithm (GA) connected with computational intelligence method is a search system used in computer science to compute approximate solutions for problems of combinatorial optimization nature. More appropriately the genetic algorithms are an optimization method built on natural evolution. They incorporate the continuity of the fittest idea algorithm [10]. In this algorithm, a population gradually converges to an ultimate individual with an associated objective value. The most desired solutions within the population are allocated the top survival rate from one generation to the next to facilitate this [11].

1.4 Simulation

A simulation is a process of imitating the procedure of a real-world practice or system. A developed model is required for the act of simulation; the key characteristics, manners and functions of the selected physical or abstract system or process are represented by this model. A computer simulation is an effort to design a real-world or hypothetical condition on a computer so that it can be studied to see how the system reacts in that condition. By altering the variables in the simulation, predictions can be made about the response of the system. It is a tool to virtually examine the behavior pattern of the system under that is being studied [12]. The major advantages of simulation come from the better understanding of interactions and identification of potential problems that simulation offers, allowing the evaluation of various substitutes and therefore, dropping the number of changes in the ultimate system [13]. Several simulation systems are in use. However, of all the techniques used to understand the impact of risk and uncertainty in financial, project management, cost, and other forecasting models, Monte Carlo simulation or probability simulation is an effective one. In a Monte Carlo simulation, a random value is chosen for each of the jobs, depending on the range of estimates. The model is simulated based on this random value. The result of the model is documented every time, and the process is repeated. A typical Monte Carlo simulation computes the model hundreds or thousands of times, each time using dissimilar randomly selected values. When the simulation is complete, there is a large number of results from the model, each based on arbitrary input values. These findings are used to describe the possibility, or probability, of reaching various results in the model.

1.5 Outline of the Thesis

The thesis is organized as follows. Chapter 1 presents an overview of warehouse functions, genetic algorithm and simulation. Chapter 2 describes the literatures related to order picking operations of warehouse and states the objectives of the thesis. In chapter 3, an efficient storage allocation method for maintenance environment warehouse is proposed. A simulation model is developed in chapter 4 to evaluate the performance of the proposed method. In chapter 5, the simulation program coded in C programming language is run and the result of the proposed storage method is compared with the other three different dedicated storage methods. The conclusion and discussion on the potential research direction is presented in chapter 6.

CHAPTER II

LITERATURE REVIEW

2.1 General

Several suitable topics are there that will be written up in the evaluation of literature, considering factors that affect the efficiency of order picking. By decreasing order pickers traveled distance, order picking efficiency can be enhanced. For improving the management of order picking four methods are usually applied such as: 1) allocating items to the precise locations; 2) zoning the warehouse; 3) batching of orders; 4) determining a route for picking [14]. The above discussed ways are firmly reliant to minimize the problem of order picking. But due to computational intractability, one or two of above-mentioned decisive sectors are emphasized by researchers.

2.2 Storage Assignment Policies

Stock keeping units (SKUs) are required to be placed into the location of store before they are ready to accomplish the request of customer. The efficiency of order picking procedure is immensely affected by storage assignment instructions [15]. Harmatuck [16] has illustrated that the COI-based item distribution can lessen the cost of expected order-picking for single-address stock location. For unit load AS/RS, Graves et al. [17] has shown that to permit an incoming load to be stored in its class region, with the increase of number of class requirements of space increase. Consequently, class-based storage needs more space in rack than randomized storage, and more rack space is required for dedicated storage compared to class-based storage. Schwarz et al. [18] examine an automated warehouse system’s performance through a particular storage policy that depends on frequency of item picking. Roll and Rosenblatt [19] made comparison of the space requirements for the random, volume-based and class-based storage project for a port warehouse by using simulation. Their outcomes show that compared to the volume-based assignment the class-based storage assignment can decrease the space requirement in a significant manner. Rosenblatt and Roll [20] studied the consequence of storage policy. They used both analytical and simulation procedures on the internal layout of warehouse. Frazelle and Sharp [21] examined the correlated assignment strategy. They proposed a heuristic technique for determining the sites of components on the basis of demand reliance between items. Park and Webster [22] suggested a class-based storage technique in a three-dimensional system. It reduces total travel periods. Malmborg and Bhaskaran [23] presented a revised proof of COI rule and assess it for various kinds of warehouses on the basis of analytical expressions. Gibson and Sharp [24] presented that finding high frequency items can expressively decrease the travel distance if they are located near to the input-output (I/O) point. Van den Berg [25] suggested a dynamic programming algorithm that dispenses items and places amongst classes in a way that considering the lessening in area the mean single command travel time is diminished due to categorizing items. Petersen et al. [26] studied the development of order picking effectiveness by forming classes on basis of grouping parts with alike COI values in a manual order picking warehouse. The performances of class-based storage are compared to both random and volume-based storage. Jane and Laih [27] developed a clustering algorithm for allocation of item by harmonizing the workload among all order pickers so that the utilization of the picking system is upgraded. De Koster et al. [6] introduced family grouping, where the products are classified according to relations or resemblances between products or orders. Mantel et al. [28] introduced a new storage allocation approach called Order Oriented Slotting (OOS), where items that appear together in orders are stored close with respect to each other. Muppani and Adil [29] developed a branch and bound algorithm to create classes and distribute storage space and to make a comparison with a benchmark dynamic programming algorithm. Their study showed that class-based policy has shorter travel distance in comparison with the dedicated policy. Bindi et al. [30] established different storage allocation laws based on the implementation of original similarity co-efficient and clustering methods. Xiao and Zheng [31] studied storage assignment problem that is correlated by considering the production bill of material (BOM) information where relatively stable order frequency is found. Accorsi et al. [32] presented a systematic hierarchical top-down method that permits the merging of successive decision steps concerned with allocation and assignment issues. Bottani et al. [33] studied genetic algorithm for optimization of item allocation with the intention of reducing travel time of pickers.

2.3 Routing Policies

The goal of routing policies is to arrangement the items on the pick list to guarantee a decent course through the warehouse. A lot of policies have been recommended that range from basic heuristics to ideal algorithm to accomplish this purpose. This issue is a warehouse-specific Traveling Salesman Problem (TSP), where the picking/stocking area of an item is given. The issue where there are several probable areas for the retrieval or capacity of an item is more complex and few research results are available, in spite of the fact that usually found practically speaking. The TSP in the warehouse is exceptional due to the aisle structure of the possible travel paths [1]. Ratliff and Rosenthal [34] suggested a dynamic programming algorithm to find out the ideal order picking course in a traditional warehouse with two cross aisles. Han et al. [35] evaluated the nearest-neighbor heuristics and the shortest-leg recovery approaches for an AS/RS which is evaluated on the basis analytical expressions and simulation. They demonstrate that the issue of ideally sequencing a given rundown of requests in NP- hard. Bachers et al. [36] compared a few traditional traveling salesman problem heuristics, for example the nearest-neighbor method, the successive insertion method, and the local search method with the help of simulation. Bozer et al. [37] proposed a routing algorithm in a two-dimensional rack with in excess of two areas to be visited and assess its execution through a simulation study. Hall [38] developed distance approximation procedures for five order picking strategies, i.e. traversal, midpoint, largest-gap, optimal routing, and minimal. Petersen [39] evaluated the execution of six heuristic routing policies and compared them to the ideal strategy in a circumstance with random storage. De Koster and Van der Poort [40] examined the issue of finding efficient order picking routes for both conventional and modern warehouses. They developed an algorithm to decide shortest order picking routes in a warehouse. Roodbergen [41] considered the routing method as traveling salesman problem and recommended six routing strategies such as: S-shape, return, mid-point, largest gap, combined and optimal. Hwang et al. [42] exhibited analytical expressions for return, S-shape and midpoint routing methods. They consider COI based storage policies for their study. Ene and Öztürk [43] developed a quicker genetic algorithm to form optimal batches and optimal routes for order pickers. The fundamental preferred standpoint of this algorithm is the quick response and can be applied to any sort of warehouse layout.

2.4 Zoning

Order picking area can be divided into zones. Instead of picking order from single place order can be picked from different areas by creating zones. In each zone there will be separate order picker. He will pick order from his respective zone. In spite of having significant impact on order-picking systems zoning problem could not gain much attention in respect of other planning issues. Advantages of creating zone comprise the fact that each order picker will cover only a smaller area, diminished traffic obstruction, and additionally crates the opportunity for order pickers to get acquainted with the locations of commodity in the zone. The foremost drawback of zoning is that before delivery to the customer split orders are needed to be combined again. To deal with this we can use two methods. The first approach should be proceeded step by step assemblage of an order. With the help of this approach one order picker begins on the order. With the completion of his part, the tote and pick list are passed to the next picker. He then carries on with the assemblage of the order. After visiting all the zones an order is completed. This process is also known as pick-and-pass. The second method used for zoning is known as parallel picking. In this process many order pickers start on the similar order, respective order picker in his corresponding zone. After picking the partial orders are combined. In implementation, zoning is partly based on product properties, like size, weight, and essential temperature and safety measures [6]. By using simulation Mellema and Smith [44] observed the effects of the aisle configuration, stocking policy, batching and zoning rules. They proposed that linkage between batching and zoning can increase the productivity. De Koster [45] presented a zoned pick-and-pass way as a Jackson queuing network. It permits rapid evaluation of order throughput times and average work-in-process. Comparison is done between outcomes and simulations. This estimation helps to determine the number of zones and the size of system. Malmborg [46] evaluated the storage policy for an AS/RS with zoning constraints, and recommends a simulated annealing algorithm, to capitalize the storage capacity. Jane [47] proposed several heuristic algorithms for balancing the workloads among the order picker and to improve the size of zone for variation of order volume in a developing zoning system. Petersen [48] illustrated that on the average travel distance within the zone the shape of zone, the number of items on the pick-list and the storage plan have a noteworthy effect. Le-Duc and De Koster [15] suggested a probabilistic model which helps to estimate the average travel distance of a picking trip. As objective function they used average travel distance. They submitted a mathematical formulation to optimize storage zone problem.

2.5 Batching

A sufficient number of orders can be collected from a single picking tour when sufficient picking capacity is available and orders are small. This method of picking is called order batching or simply batching. It is a process in which a set of orders are grouped into sub-sets, and afterwards retrieved in a single picking tour [49]. Batching of order is a problem of NP-hard type. For this reason, many researchers emphasize on developing heuristic methods to solve it. SL algorithm (combine Small with Large orders) is one of the four batching heuristics proposed by Elsayed and Unal [50]. It differentiates between small and large orders and minimizes travel distance. On the basis of average travel times, a comparative study of batching heuristics was performed by Pan and Liu [51]. The shape and the capacity of the cranes have been tested as well as the variation of storage assignment policies have been made. The authors concluded that only the storage capacity of a crane has a great impact on the choice for a certain law. Elsayed et al. [52] and Elsayed and Lee [53] considered the order-batching problem in a man-aboard system with a view to minimizing the penalties and tardiness as corresponding purposes. Their proposed heuristic creates batches at first and then measures the release times for the batches. Ruben and Jacobs [54] developed batch construction heuristics. They found that on the order retrieval efforts in a warehouse, the procedures used for creating batches of orders and for assigning storage space to distinct items have a great impact. Gademann and Van de Velde [55] studied the order-batching problem in order to lessen the total travel time. They illustrated that when the number of orders per batch is more than two this type of problem is still considered NP-hard. In order to solve instances of modest size to optimality an algorithm called branch-and-price is designed. In case of larger instances, it is recommended to use an iterated descent approximation algorithm. Won and Olafsson [56] concentrate on response time of customer by taking into consideration of batching and picking operation.

2.6 Thesis Objectives

It is evident from the above discussion, most of the researchers focus on distribution centers or retail warehouses, where order can be changed dynamically. For this reason, they use similarity coefficient to identify correlations between items. On the other hand, maintenance environment warehouse is very similar to production warehouse where order structures are limited. In maintenance environment spare parts required for a particular type of maintenance is heterogeneous but they are similar in the sense that they are used for a specific maintenance purpose. For this reason, most of the time a particular maintenance requires spare parts from a predefined set of spare parts. So, it should be better approach to apply group technology which is an approach that seeks to identify those attributes of a population that permit its members to be collected into groups, sometimes called families. The members of each particular group possess attributes that are similar. If it is possible to group spare parts on the basis of similarity of usages and locate near each other, then the retrieval of parts will be easier and less time consuming. Again, the COI rule has been considerably studied in the literature and proved that it is optimal in minimizing the material handling cost in dedicated storage. So, it will be efficient if the groups with minimum COI are assigned near to the I/O point. Hence, the objectives of the thesis are as follows:

(1) To develop an efficient storage allocation method.

(2) To evaluate the performance of the proposed storage method using simulation.

CHAPTER III

RESEARCH METHODOLOGY

3.1 System Description

The problem that is considered for this study is a maintenance type warehouse. Spare parts are stored and retrieved for maintenance purposes in a maintenance type warehouse. In such an environment, allocation of space and assignment of spare-parts is a vital problem for smooth operation in a warehouse. Miss-allocation of spare-parts takes huge time to retrieve specific parts. The demand for spare parts for a particular maintenance work depends upon the type of maintenance work. So, the spare parts required for a maintenance work is a sub set of a super set. This super set can be determined by obtaining the list of all the spare parts required for the particular maintenance work. This super set is defined as spare part set in this study. For instance, spare parts essential for maintenance of injection pump and engine head of internal combustion engine might be dissimilar in type and in amount. The spare part set for these two types of maintenance work hardly have common intersection. If it is possible to identify spare part set required for a particular type of maintenance work, then the set of parts with lowest COI can be kept in one region closest to the I\O point so that the searching and retrieval time can be minimized. The following assumptions [57] are made in this study:

- A single-sided storage rack with 42 storage locations, as shown in figure 2, is considered in which the I/O point is located at the extreme lower left-hand corner.
- Each storage location (or shelf) is uniform in size (2ft × 2ft × 2ft) and reserved for a specific type of spare parts i.e., dedicated storage.
- The storage or retrieval (S/R) device can travel simultaneously in both the vertical and the horizontal directions at constant velocity.

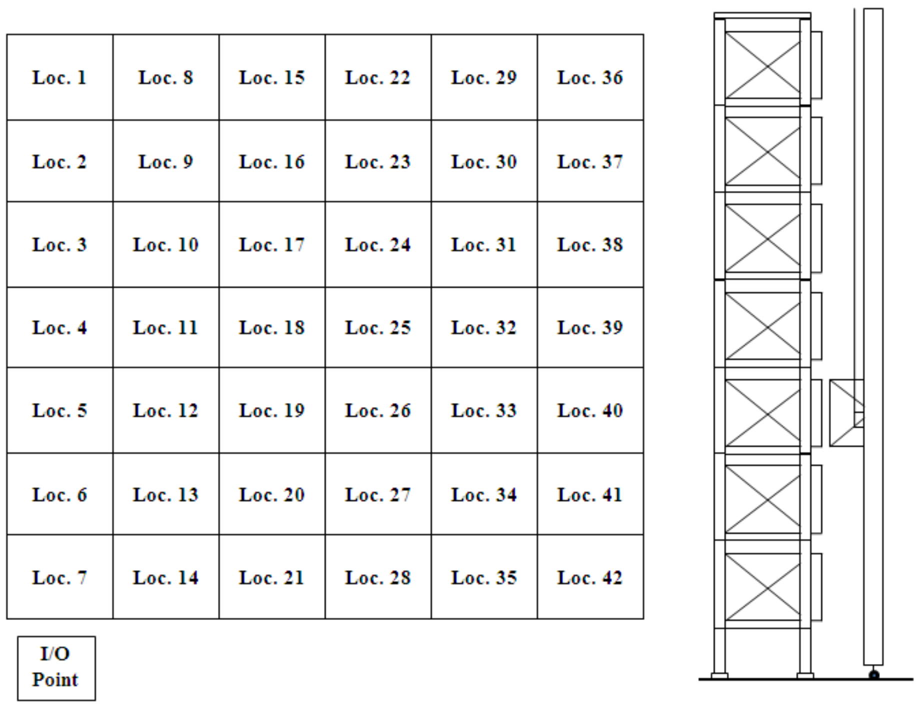


Figure 2: Front view and side view of the warehouse rack [58]

3.2 Proposed Storage Method

The proposed storage method consists of five following stages as shown in figure 3.

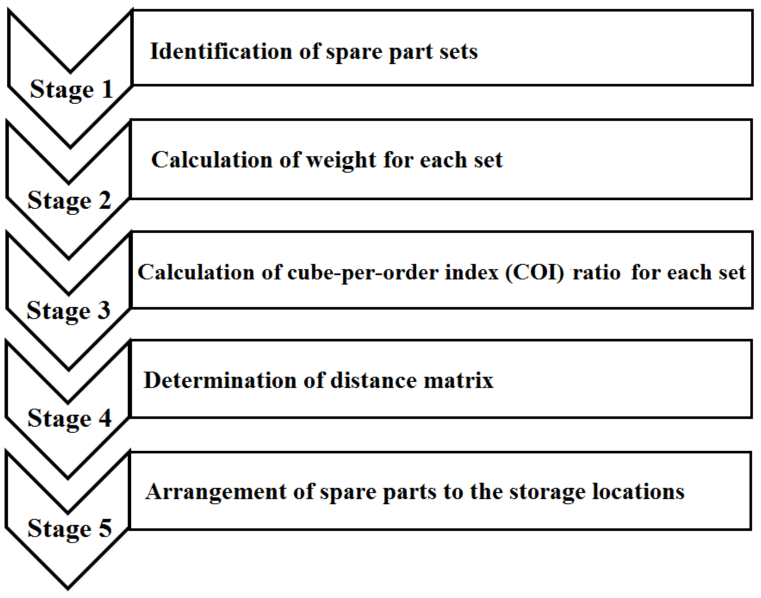

Stage 1
Identification of spare part sets
Stage 2
Calculation of weight for each set
Stage 3
Calculation of cube-per-order index (COI) ratio for each set
Stage 4
Determination of distance matrix
Stage 5
Arrangement of spare parts to the storage locations

Figure 3: Methodology of proposed storage allocation

Stage 1: Identification of spare part sets

The demand for spare parts for a particular type of maintenance can be found from the issue register of the warehouse. Let us assume that

m = type of maintenance, $m = 1, 2, 3, \ldots, M$

t = time of maintenance, $t = 1, 2, 3, \ldots, T$

D_{mt} = the demand set is the set of spare parts demanded for maintenance type 'm' at time period 't'

S_m = set of spare parts for maintenance type 'm'

From these past demand sets, the sets of spare parts can be determined with the following formula:

$$S_m = \bigcup_{t=1}^{T} D_{mt}$$

Here, if a part is appeared in more than one group then the part will be in the group where the demand frequency is higher.

Stage 2: Calculation of weight for each set

The weight is used to identify a set as fast moving or slow moving. As the fast moving sets are used more frequently so these sets should be kept near to the issues counter as opposed to the slow moving sets. A simple way to determine the weight of a set could be the sum of annual usage frequency of all parts in the set. Let,

N_m= Total number of spare parts in set 'S_m'

f_{im}= Annual usage frequency of spare part 'i' in set S_m, $i = 1, 2, 3, \ldots, N_m$

So, the weight of set 'S_m' will be

$$W_m = \sum_{i=1}^{N_m} f_{im}$$

Stage 3: Calculation of cube-per-order index (COI) value for each set

As per our assumption, each spare part is stored into a specific shelf. So, the total number of shelves required for a set is equal to the number of spare parts in the set. Let, the volume of a shelf is 'v'. Therefore, the total volume required for a set will be,

$$V_m = N_m \times v$$

Now, cube-per-order index (COI) value for a set 'S_m' will be,

$$COI_m = \frac{V_m}{W_m}$$

Stage 4: Determination of distance matrix

Let us assume that, the distance between I/O point and the most nearest shelf is 3 meter and the distance between any two adjacent shelves is 2 meter. According to the assumption, the S/R device can travel to the vertical and the horizontal directions simultaneously. So, the distance between any two locations i and j (including the I/O point) in two dimensional co-ordinate is given by the following formula:

$$d_{i,j} = |x_i - x_j| + |y_i - y_j|$$

Finally, the distance values are stored in a distance matrix. For example, the distance between I/O point and location 33 (see figure 4) is,

$$d_{0,33} = 8+7 = 15 \text{ ft.}$$

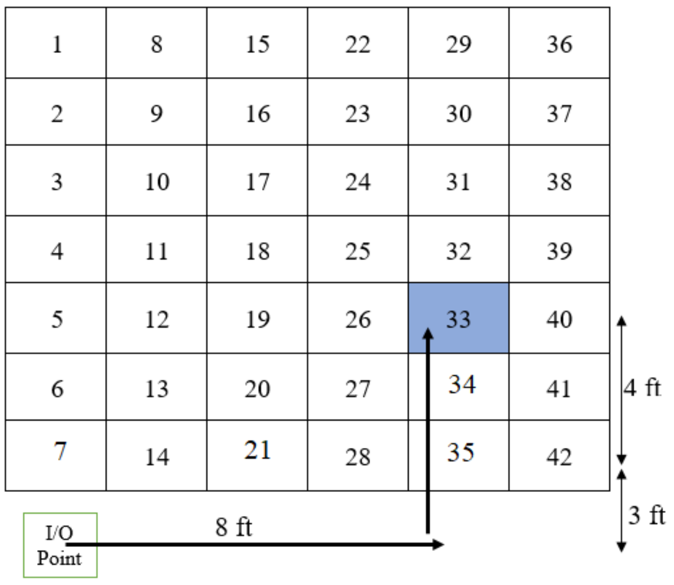


Figure 4: Distance between I/O point to location 33

Stage 5: Arrangement of spare parts to the storage locations

Here, the sets are assigned to storage locations according to COI policy. Since each type of parts occupies the same amount of space, the frequency rule will be the same as the COI rule. So, the spare parts within each set are arranged in descending order of frequency. The proposed storage algorithm is given below:

Step 1. Arrange the shelf numbers in ascending order of distance from I/O point (name this list as *A*).

Step 2. Arrange the sets in ascending order of COI value (name this list as *B*).

Step 3. Take a set from the top of the list *B*.

Step 4. Arrange the parts of this set in descending order of frequency.

Step 5. Assign the parts into the shelves from the top of the list *A*.

Step 6. Continue steps 3 to 5 until the list *B* is empty.

CHAPTER IV

DEVELOPMENT OF SIMULATION MODEL

4.1 Experimental Design

A single-sided storage rack is considered for the experiment to evaluate the performance of the proposed storage method. The storage rack is considered due to its simplicity and to avoid the computational complexity. The simulation model assumes the pattern of a Monte Carlo process. Furthermore, it is assumed that there are 7 types of maintenance works and all the spare parts are required by these maintenance works. Total number of parts in each spare part set varied from 4 to 8. The usage frequency of a spare part for a particular maintenance work is varied from 8 to 72 in a year. Table 1 represents the assumed sets of spare parts and the corresponding frequency sets. The frequency set shows the annual usage frequency of each part in the set. The weight of a set is the sum of usage frequency of all parts in the frequency set. In the simulation model a maintenance type is selected according to the weight. Table 2 shows the random number generation scheme for selecting the types of maintenance. Similarly, spare parts from a set are selected according to their demand frequency. Table 3 shows the random number generation scheme for selecting spare parts from the set S_1 related to type-1 maintenance.

Table 1**:** Weight and COI calculation of each set of spare parts

Maintenance type, *m*	Spare part set (1-42), S_m	Frequency set	Weight (W_m)	COI_m
1	[6,7,15,28,36,42]	[8,47,52,30,21,38]	196	0.245
2	[9,14,16,18,22,24,32]	[62,19,50,54,36,57,9]	287	0.195
3	[5,21,25,34,41]	[23,32,65,27,47]	194	0.206
4	[1,11,13,26,31,35]	[53,58,26,32,14,41]	224	0.214
5	[2,3,4,8,10,17,20,40]	[18,52,43,16,20,23,25,34]	231	0.277
6	[12,19,29,33]	[21,26,15,30]	92	0.348
7	[23,27,30,37,38,39]	[72,26,53,41,58,30]	280	0.171

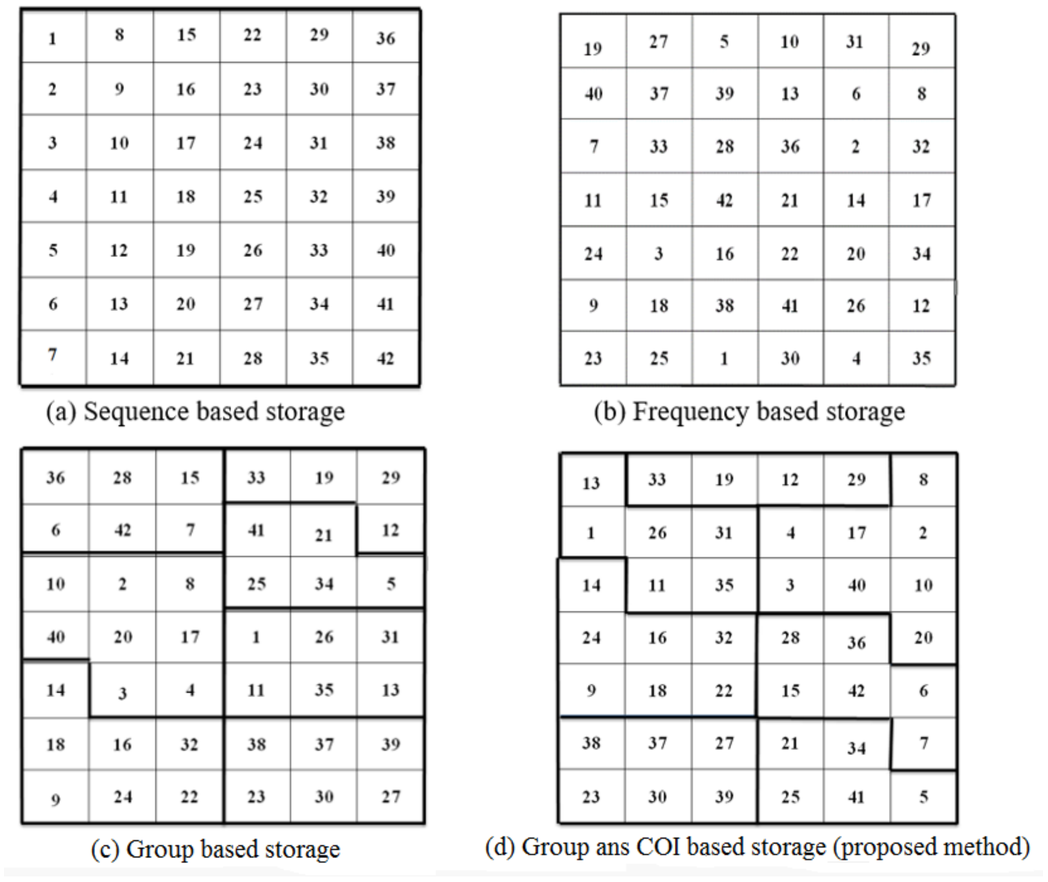
Table 2: Random number generation scheme for selecting types of maintenance

Maintenance type, *m*	Weight	Probability	Cumulative	Range
1	196	196/1504=0.130	0.130	0<R≤0.130
2	287	0.191	0.321	0.130<R≤0.321
3	194	0.129	0.450	0.321<R≤0.450
4	224	0.149	0.599	0.450<R≤0.599
5	231	0.154	0.753	0.599<R≤0.753
6	92	0.061	0.814	0.753<R≤814
7	280	0.186	1	0.814<R≤1

Table 3: Random number generation scheme for selecting spare parts from the set S_1

Spare part No.	Demand frequency	Probability	Cumulative	Range
6	8	8/196=0.041	0.041	0<R≤0.041
7	47	0.240	0.281	0.041<R≤0.281
15	52	0.265	0.546	0.281<R≤0.546
28	30	0.153	0.699	0.546<R≤0.699
36	21	0.107	0.806	0.699<R≤0.806
42	38	0.194	1	0.806<R≤1

Four dedicated storage allocation methods including the proposed storage method are considered for the comparative analysis. The first one is sequence based storage, in which the spare parts are stored according to part number sequence i.e., alphanumeric order without regard to order frequency, size, weight etc. This method is often used due to its simplicity [59]. The second one is frequency based or volume based storage, in which spare parts are stored according to the annual usage frequency of each part. In this method, the parts with the largest demand are kept closest to the I/O point [60]. The third one is group based storage, in which spare parts are stored according to the group or similarity of usage. This method only consider the total order frequency of each group, not their space occupation, for the assignment to storage locations [61]. The fourth one is group and COI based storage, which is the proposed method developed in this research. The proposed method consider both the demand frequency and the space occupation of each group. These four storage allocation methods (see figure 5) are simulated for the single-sided storage rack to investigate the performance of the proposed storage method in terms of average travel distance of order pickers.



1	8	15	22	29	36
2	9	16	23	30	37
3	10	17	24	31	38
4	11	18	25	32	39
5	12	19	26	33	40
6	13	20	27	34	41
7	14	21	28	35	42

(a) Sequence based storage

19	27	5	10	31	29
40	37	39	13	6	8
7	33	28	36	2	32
11	15	42	21	14	17
24	3	16	22	20	34
9	18	38	41	26	12
23	25	1	30	4	35

(b) Frequency based storage

36	28	15	33	19	29
6	42	7	41	21	12
10	2	8	25	34	5
40	20	17	1	26	31
14	3	4	11	35	13
18	16	32	38	37	39
9	24	22	23	30	27

(c) Group based storage

13	33	19	12	29	8
1	26	31	4	17	2
14	11	35	3	40	10
24	16	32	28	36	20
9	18	22	15	42	6
38	37	27	21	34	7
23	30	39	25	41	5

(d) Group ans COI based storage (proposed method)

Figure 5: Four dedicated storage allocation methods

4.2 Traveling Distance Calculation

Travel distance refers to the distance between the I/O point from where the S/R device has to start travel to collect parts from the shelves and come back to that point. To collect 'n' numbers of parts from the rack with minimum travel distance is a traveling salesman problem (TSP). We assumed that the S/R device picks only one order at a time. The total distance traveled to collect 'n' parts of an order is given by the following formula:

$$d = d_{0,i} + d_{i,j} + d_{j,k} + d_{k,l} + \dots\dots\dots + d_{(n-1),n} + d_{n,0}$$

Figure 6 shows how to determine the distance travelled by the picking device to collect spare parts 5, 21, 25, 34, 41.

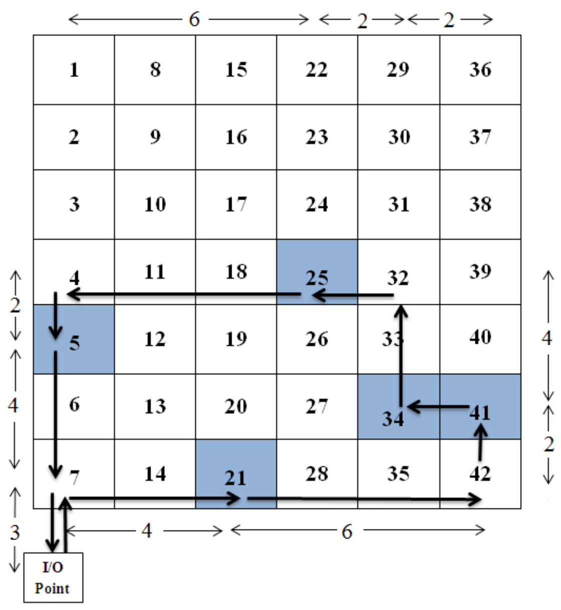


Figure 6: Sample travel distance calculation

So, the total distance travelled, d = 3+4+6+2+2+4+2+6+2+4+3 = 38 ft.

4.3 Genetic Algorithm for Routing Problem

Finally, to solve the TSP of each tour, the well-known genetic algorithm is used which is a very good global search technique. The algorithm is embedded into the simulation program to estimate the optimal traveling sequence to minimize the total travel distance to collect all the parts of a demand. For this purpose, we incorporate the distance matrix of the locations into the algorithm. At first, the algorithm generates a preset number of random tours and then improves the population until a stop condition is satisfied and the best tour is returned as the solution. Order crossover technique [62] is used in this algorithm, illustrated in figure 7. In this technique, one segment from parent 2 is selected randomly and copied into child 1 in the same position. The rest portion of the child 1 will be filled from parent 1 sequentially except the elements which have already occupied. Similarly, child 2 will be generated in the same manner. Moreover, exchange mutation operation [63] has been applied in the genetic algorithm, described in figure 8. In this operation, two randomly selected elements in the child are swapped with each other. We also initialize the population size according to the number of locations to be traveled.

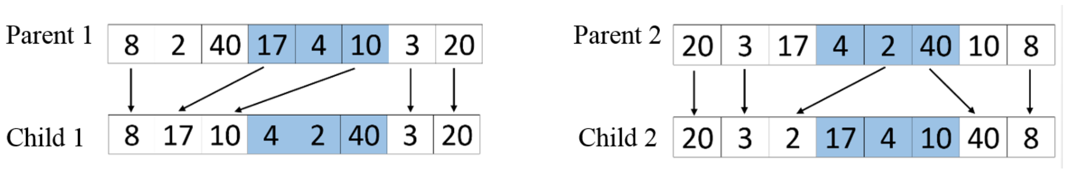


Figure 7: An example of Order Crossover

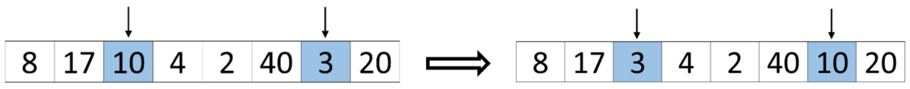


Figure 8: An example of Exchange Mutation

The pseudo-code of the genetic algorithm is given below:

```
(1)  Generate initial population with a number of random travel routes;
(2)  **For** each route (chromosome)
(3)       Evaluate its fitness value (length of travel);
(4)  **Next**
(5)  Rank all the routes according to the fitness value;
(6)  Select predefined number (20% of population) of best routes for new population
     (elitism);
(7)  Select two distinct routes by applying roulette wheel mechanism;
(8)  Apply order crossover technique to generate two new routes (child) for new
     population;
(9)  Apply exchange mutation operation for each child with probability of 0.01.
(10) **If** the size of new population is not being completed **then**
(11)      Return to step (7);
(12) **For** each modified route
(13)      Evaluate its fitness value;
(14) **Next**
(15) **If** the stop condition is not reached **then**
(16)      Return to step (5);
(17) Terminate and return the optimal route and its length for a demand;
```

The pseudo-code of the simulation program is given below:

```
(1) **Input** the total number of travels $N$;
(2) **For** travel no. = 1 to $N$
(3)     Randomly generate a maintenance type ‘$m$’ according to their weight;
(4)     Randomly generate number of parts ‘$n$’ required for the maintenance work
        according to uniform probability distribution;
(5)     Randomly select the ‘$n$’ number of parts one by one from the set of spare parts
        according to their demand frequency;
(6)     Find the optimal traveling sequence and distance by using genetic algorithm for
        different storage methods;
(7) **Next**
(8) Evaluate average travel distance for different storage methods;
(9) Terminate the simulation;
```

CHAPTER V

SIMULATION RESULTS

5.1 Results

The computer simulation program for the experiment was coded in C programming language. C language is much easier than any other language in implementing complex algorithms. The simulation program performs on the basis of randomly generated data as per our assumption. The simulation terminates once a user-defined number of iterations (travels) has been completed. The iteration number of the simulation is significantly large enough due to the stochastic nature of the process. The data are generated from uniform probability distribution within specified limits. At first, we run the simulation program with the length of 100 travels performed on the randomly generated maintenance types. Table 4 shows the resulting average travel distance from each travel for the proposed method of storage. Similarly, Table 5 shows the optimum traveling sequence and the corresponding travel distance for other three dedicated storage methods.

Table 4: Simulation results for the storage with proposed method

No	**Maintenance type, *m***	**Number of spare parts required, *n***	**Spare parts to be picked**	**Traveling sequence by Genetic algorithm**	**Travel distance (ft)**	**Average travel distance (ft)**
1	4	5	1 35 11 26 31	11 35 31 26 1	34	34
2	2	5	24 9 16 14 22	22 16 14 24 9	30	32
3	7	3	23 39 37	23 37 39	18	27.3
4	5	4	40 2 17 3	3 2 17 40	46	32
5	4	6	11 13 1 26 35 31	13 1 26 31 35 11	38	33.2
6	2	3	16 18 22	16 22 18	26	32
7	6	2	33 29	29 33	46	34
8	4	3	11 35 1	1 11 35	34	34
9	7	1	23	23	6	30.9
10	4	4	26 1 13 35	13 1 26 35	38	31.6
11	6	2	29 33	33 29	46	32.9
12	1	4	7 28 36 15	15 28 36 7	38	33.3
13	5	4	40 2 10 17	17 2 10 40	46	34.3

14	7	4	39 23 30 27	23 39 27 30	18	33.1
15	2	4	22 32 14 16	22 32 16 14	30	32.9
16	6	3	33 12 29	33 29 12	46	33.8
17	1	1	15	15	26	33.3
18	1	5	7 42 28 15 36	7 42 36 28 15	38	33.6
19	6	2	33 29	29 33	46	34.2
20	4	6	11 13 26 1 35 31	13 1 26 31 35 11	38	34.4
21	4	3	11 1 13	11 13 1	34	34.4
22	5	7	4 40 8 20 3 17 10	3 40 20 10 8 17 4	54	35.3
23	1	4	42 7 36 28	28 36 42 7	38	35.4
24	6	3	33 12 19	33 19 12	42	35.7
25	3	5	25 34 41 21 5	25 21 34 41 5	30	35.4
26	2	2	18 9	9 18	18	34.8
27	6	4	33 19 29 12	33 19 12 29	46	35.2
28	4	2	26 1	1 26	30	35
29	6	4	33 12 19 29	33 12 29 19	46	35.4
30	2	7	24 9 18 16 22 14 32	9 18 22 32 16 14 24	30	35.2
31	7	3	37 38 39	39 37 38	18	34.6
32	7	2	39 30	30 39	14	34
33	4	5	26 1 31 35 11	11 35 31 26 1	34	34
34	7	6	38 30 39 27 37 23	23 30 39 27 37 38	18	33.5
35	5	2	2 20	20 2	46	33.9
36	3	4	25 41 34 21	21 34 41 25	26	33.7
37	3	3	41 21 25	41 25 21	26	33.5
38	2	7	24 9 18 16 22 14 32	18 22 32 16 14 24 9	30	33.4
39	2	3	24 18 9	24 9 18	22	33.1
40	7	4	23 38 37 27	23 38 27 37	18	32.7
41	2	5	22 18 16 9 14	9 22 18 16 14	30	32.6
42	7	2	38 39	39 38	18	32.3
43	5	4	40 3 17 8	3 40 8 17	50	32.7
44	2	2	9 24	24 9	18	32.4
45	3	4	5 25 21 41	25 5 41 21	30	32.3
46	6	4	29 19 33 12	19 29 12 33	46	32.6
47	4	6	35 1 11 13 31 26	1 13 26 31 35 11	38	32.7
48	2	5	16 9 18 24 22	9 18 22 16 24	26	32.6
49	3	4	41 25 21 34	25 41 34 21	26	32.4
50	5	4	4 8 40 20	4 8 20 40	54	32.9
51	3	1	25	25	18	32.6

52	1	4	15 42 6 7	7 6 42 15	34	32.6
53	3	2	41 21	21 41	26	32.5
54	1	6	42 6 36 7 28 15	15 28 36 42 6 7	38	32.6
55	7	2	27 38	38 27	18	32.3
56	1	1	42	42	30	32.3
57	6	4	33 12 19 29	33 12 29 19	46	32.5
58	5	7	10 4 17 40 3 20 2	20 10 2 4 17 40 3	50	32.8
59	7	1	23	23	6	32.4
60	3	1	25	25	18	32.1
61	4	2	11 26	26 11	30	32.1
62	7	2	23 37	37 23	14	31.8
63	1	6	15 7 28 42 36 6	7 6 42 36 28 15	38	31.9
64	1	5	7 42 6 36 15	15 36 42 6 7	38	32
65	7	1	39	39	14	31.7
66	1	1	15	15	26	31.6
67	1	4	36 7 28 15	15 28 36 7	38	31.7
68	2	1	18	18	18	31.5
69	4	4	13 1 26 11	11 26 1 13	34	31.6
70	7	3	30 23 37	30 37 23	14	31.3
71	4	4	1 35 13 11	35 11 1 13	38	31.4
72	2	3	14 22 18	14 18 22	30	31.4
73	7	4	37 38 23 39	38 37 39 23	18	31.2
74	5	3	17 40 3	3 17 40	42	31.4
75	5	5	4 40 3 17 10	3 40 10 17 4	46	31.5
76	2	4	24 18 9 16	18 16 24 9	22	31.4
77	3	1	34	34	26	31.4
78	7	2	23 30	30 23	10	31.1
79	5	4	40 2 4 3	3 4 2 40	46	31.3
80	2	1	16	16	22	31.1
81	7	1	23	23	6	30.8
82	2	5	14 18 24 22 32	18 22 32 14 24	30	30.8
83	3	2	25 5	5 25	26	30.8
84	1	3	7 36 6	7 6 36	38	30.9
85	2	2	18 16	16 18	22	30.8
86	4	2	1 26	26 1	30	30.7
87	5	4	4 3 20 17	3 4 17 20	46	30.9
88	3	4	5 25 34 21	25 21 34 5	30	30.9
89	3	4	41 25 34 21	25 21 34 41	26	30.9
90	2	4	22 9 14 16	9 22 16 14	30	30.8
91	7	5	23 27 39 30 37	39 27 37 30 23	18	30.7
92	1	6	7 28 15 36 42 6	15 28 36 42 6 7	38	30.8

93	2	1	16	16	22	30.7
94	3	4	41 25 21 34	25 41 34 21	26	30.6
95	6	2	19 33	33 19	38	30.7
96	6	3	29 33 19	29 19 33	46	30.9
97	2	4	18 32 16 22	22 32 16 18	26	30.8
98	3	1	34	34	26	30.8
99	4	5	35 11 1 26 31	1 26 31 35 11	34	30.8
100	2	6	18 22 24 9 16 14	9 24 14 16 18 22	30	30.8

Table 5: Simulation results for the other three dedicated storage methods

No	**Sequence Based Storage**		**Frequency Based Storage**		**Group Based Storage**	
	Traveling sequence by GA	Travel distance (ft)	Traveling sequence by GA	Travel distance (ft)	Traveling sequence by GA	Travel distance (ft)
1	11 1 31 26 35	54	11 31 26 35 1	50	11 35 31 26 1	38
2	9 16 22 24 14	42	9 16 22 14 24	34	9 24 22 16 14	22
3	39 37 23	46	23 37 39	34	23 39 37	30
4	3 2 17 40	46	3 17 2 40	46	40 2 17 3	30
5	35 26 31 1 11 13	54	11 13 31 26 35 1	50	11 1 26 31 13 35	38
6	18 16 22	42	16 22 18	26	18 16 22	18
7	29 33	46	29 33	50	29 33	50
8	35 11 1	46	11 1 35	38	11 35 1	34
9	23	38	23	6	23	18
10	1 13 26 35	50	1 35 26 13	46	1 26 13 35	38
11	33 29	46	33 29	50	33 29	50
12	7 28 36 15	50	15 28 36 7	34	36 28 15 7	38
13	2 10 17 40	46	17 2 10 40	50	10 2 17 40	30
14	23 30 39 27	46	23 27 39 30	42	23 30 39 27	30
15	14 16 22 32	46	16 32 14 22	42	14 16 32 22	22
16	12 33 29	46	33 29 12	50	12 29 33	50
17	15	38	15	22	15	38
18	15 36 42 28 7	50	7 36 28 42 15	34	36 28 15 7 42	38
19	29 33	46	29 33	50	29 33	50
20	13 26 35 31 1 11	58	11 13 31 26 35 1	50	35 13 31 26 1 11	38
21	1 11 13	34	11 13 1	38	11 1 13	38
22	20 40 17 8 10 3 4	50	3 40 10 8 17 20 4	54	20 3 4 17 8 10 40	34
23	7 42 36 28	50	7 28 36 42	34	36 28 7 42	38
24	12 33 19	30	19 33 12	50	12 19 33	50
25	5 25 34 41 21	38	25 5 21 34 41	50	25 34 5 21 41	46
26	9 18	34	9 18	14	9 18	10
27	19 33 29 12	46	12 29 19 33	54	33 19 29 12	50

28	1 26	42	1 26	26	1 26	34
29	12 19 33 29	46	12 29 19 33	54	33 19 29 12	50
30	14 32 24 22 9 16 18	50	9 18 16 22 32 14 24	42	9 14 18 16 32 22 24	22
31	37 38 39	46	37 39 38	34	38 37 39	30
32	30 39	46	30 39	38	30 39	30
33	1 11 26 31 35	54	1 35 26 31 11	50	35 31 26 1 11	38
34	23 30 37 38 39 27	46	30 38 39 27 37 23	42	23 38 37 39 27 30	30
35	20 2	34	20 2	38	20 2	26
36	25 41 34 21	38	25 41 21 34	38	34 21 41 25	42
37	21 41 25	38	25 41 21	30	41 21 25	42
38	18 32 24 22 16 9 14	46	9 18 16 22 32 14 24	42	9 14 18 16 32 22 24	22
39	18 24 9	38	24 9 18	18	24 18 9	14
40	38 37 23 27	46	23 38 27 37	38	23 38 37 27	30
41	18 22 16 9 14	42	9 16 22 14 18	34	9 22 16 18 14	22
42	39 38	42	39 38	34	39 38	30
43	40 17 8 3	50	40 8 17 3	46	3 17 8 40	30
44	24 9	38	24 9	14	24 9	10
45	21 41 25 5	38	25 41 21 5	42	25 5 21 41	46
46	19 29 33 12	46	33 19 29 12	54	12 29 19 33	50
47	35 26 31 1 11 13	54	1 35 26 31 13 11	50	11 35 13 31 26 1	38
48	18 24 22 16 9	42	24 16 22 18 9	26	24 22 16 18 9	18
49	21 25 34 41	38	25 41 34 21	38	25 41 21 34	42
50	4 8 40 20	50	40 8 20 4	46	40 20 8 4	30
51	25	30	25	10	25	34
52	42 15 6 7	50	7 6 42 15	42	6 42 7 15	38
53	21 41	30	21 41	30	21 41	42
54	7 6 15 36 42 28	50	7 6 36 28 42 15	42	6 36 28 15 7 42	38
55	38 27	42	38 27	38	38 27	30
56	42	26	42	26	42	30
57	12 19 33 29	46	12 29 19 33	54	33 19 29 12	50
58	20 40 17 10 3 2 4	46	3 40 10 2 17 20 4	54	3 4 17 20 2 10 40	30
59	23	38	23	6	23	18
60	25	30	25	10	25	34
61	26 11	30	26 11	34	26 11	34
62	37 23	46	37 23	30	37 23	26
63	7 28 42 36 15 6	50	15 42 28 6 36 7	42	42 7 15 28 36 6	38
64	7 42 36 15 6	50	15 42 6 36 7	42	6 36 15 7 42	38
65	39	38	39	34	39	30
66	15	38	15	22	15	38
67	7 28 36 15	50	15 28 36 7	34	36 28 15 7	38
68	18	26	18	14	18	10
69	13 26 11 1	42	1 26 13 11	42	1 26 13 11	38

70	23 30 37	46	30 37 23	38	23 30 37	26
71	35 13 11 1	46	11 13 35 1	46	1 35 13 11	38
72	14 22 18	42	18 14 22	34	22 14 18	22
73	23 37 38 39	46	38 39 37 23	34	23 38 39 37	30
74	40 17 3	42	3 17 40	46	40 17 3	26
75	40 17 10 3 4	42	4 17 10 40 3	54	10 40 17 4 3	30
76	9 16 24 18	38	18 16 24 9	22	9 18 16 24	14
77	34	26	34	34	34	38
78	30 23	42	30 23	18	30 23	22
79	4 2 3 40	46	4 2 40 3	46	40 2 4 3	30
80	16	34	16	22	16	14
81	23	38	23	6	23	18
82	14 18 24 22 32	46	24 22 14 32 18	42	14 18 32 22 24	22
83	5 25	30	5 25	38	5 25	42
84	7 36 6	50	7 6 36	42	7 36 6	38
85	16 18	34	16 18	22	16 18	14
86	26 1	42	26 1	26	26 1	34
87	20 17 3 4	30	3 17 20 4	38	3 20 17 4	26
88	5 25 34 21	34	25 5 21 34	50	25 5 34 21	46
89	25 41 34 21	38	25 41 21 34	38	25 34 21 41	42
90	14 9 16 22	42	9 16 22 14	34	9 14 16 22	22
91	39 37 30 23 27	46	23 37 27 39 30	42	37 39 27 30 23	30
92	15 36 42 28 6 7	54	7 36 6 28 42 15	42	6 42 7 15 28 36	38
93	16	34	16	22	16	14
94	21 25 34 41	38	25 41 34 21	38	25 41 21 34	42
95	33 19	30	33 19	34	33 19	46
96	29 33 19	46	19 29 33	50	33 29 19	50
97	16 22 32 18	46	18 32 22 16	42	22 32 16 18	18
98	34	26	34	34	34	38
99	1 11 26 31 35	54	1 35 26 31 11	50	1 31 26 35 11	38
100	9 16 22 24 18 14	42	9 24 22 14 16 18	34	14 18 16 22 24 9	22

5.2 Comparison of Results

It can be seen from figure 9, the average travel distance of picking device for all the storage methods become almost stable. It also shows that, the storage method proposed in this research has the lowest average travel distance which is 30.80 feet. On the other hand, the average distances travelled for sequence, frequency and group based storages are 42.36, 36.96, and 32.56 feet respectively.

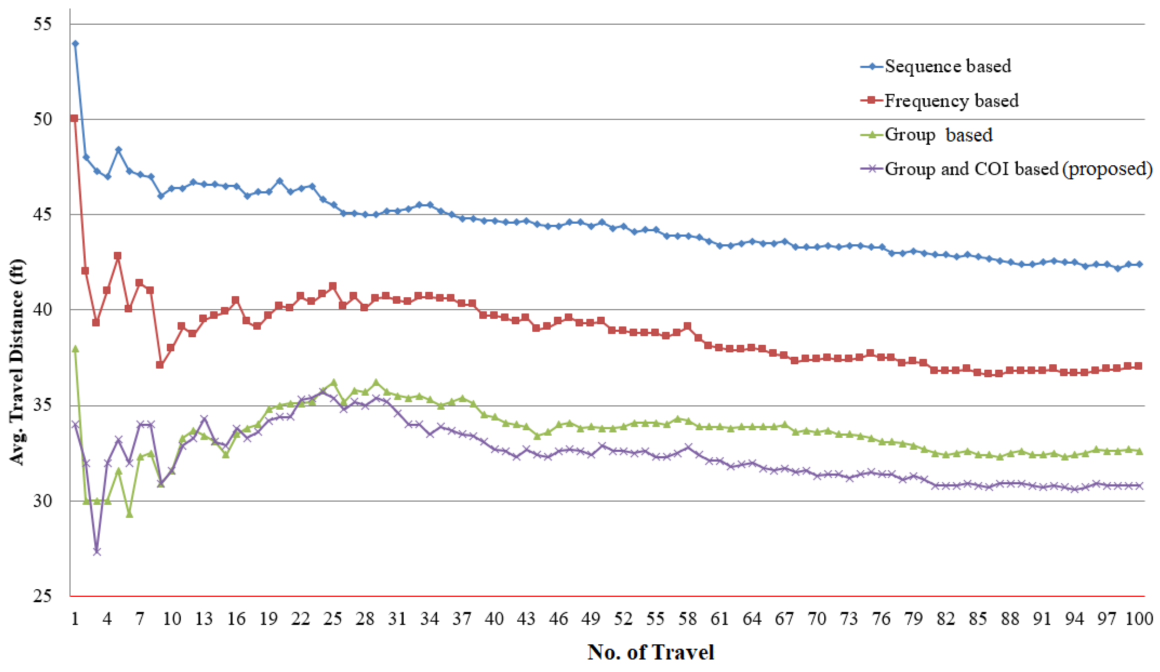


Figure 9: Comparison of average travel distance for various storage methods

Finally, we run the simulation program for ten times with the length of 1000 travels each, which is very large enough to ensure that the results depend on the given probability distribution rather than the choice of random numbers. The average execution time needed to run this length is 53.34 second. In this case, the result shows that the proposed storage allocation method has the highest reduction in the travel distance for each replication. Table 6 represents the percentage of travel distance reduction of proposed storage method with respect to other storage allocation methods. It has around 24.59% reduction in travel distance with respect to sequence based storage. Similarly, it has 18.07% and 2.96% reduction with respect to frequency and group based storage respectively.

Table 6: Simulation results for 10 independent replications

Replication	Travel distance reduction (%) of proposed storage method with respect to		
	Sequence based	Frequency based	Group based
1	25.58	17.98	3.47
2	24.97	17.89	2.90
3	25.14	18.06	3.70
4	22.94	17.38	2.16
5	24.62	18.18	2.61
6	25.23	18.63	3.43
7	23.98	17.47	2.35
8	23.35	18.73	3.38

9	25.39	18.34	3.17
10	24.70	18.07	2.43
Average Reduction (%)	24.59	18.07	2.96

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

Warehouse managers are concerned about searching for the most effective way of managing order picking, minimizing the costs involved in terms of travel distance. For maintenance environment warehouse, it is possible to group spare parts according to the type of maintenance works. The group with the lowest COI ratio can be kept closest to the I\O point. Moreover, the parts of a group can be arranged according to their usage frequency. In the simulation study, it is found that the proposed method has the highest reduction in the distance travelled by the picking device in the warehouse compared to the other storage methods. So, the proposed storage method can be applied to any production warehouse where the order structures are limited. On the other hand, the storage method will be easy to implement and simple to understand by a warehouse manager when it is adopted.

6.2 Scopes for future research

For this research, a single-sided rack was considered only. Therefore, the developed storage allocation model can be extended for multi rack warehouse system and figuring out the effects on the order picking efficiency. It is also suggested to considering different warehouse layout and incorporating more realistic issues that would be very interesting but challenging problem. On the other hand, the routing policy was restricted to follow the Genetic algorithm developed in this research. Thus, it would be interesting to use different routing policies and observe the effect on the total travel distance. Furthermore, it is also recommend to consider items that are frequently ordered together in multi item orders could be stored near each other in the super set which is also known as correlated assignment.

REFERENCES

[1] Gu, J., Goetschalckx, M., & McGinnis, L. F. (2007). Research on warehouse operation: A comprehensive review. European journal of operational research, 177(1), 1-21.

[2] Tompkins, J.A., White, J.A., Bozer, Y.A., Frazelle, E.H. and Tanchoco, J.M.A. (2003). Facilities Planning (NJ: John Wiley & Sons).

[3] Sooksaksun, N. (2012). Pareto-based multi-objective optimization for two-block class-based storage warehouse design. Industrial Engineering and Management Systems, 11(4), 331-338.

[4] Hausman, W. H., Schwarz, L. B., & Graves, S. C. (1976). Optimal storage assignment in automatic warehousing systems. Management science, 22(6), 629-638.

[5] Eckrot, A., Geldhauser, C., & Jurczyk, J. (2017). A simulated annealing approach to optimal storing in a multi-level warehouse. arXiv preprint arXiv:1704.01049.

[6] De Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. European journal of operational research, 182(2), 481-501.

[7] Heskett, J. L. (1963). Cube-per-order index-a key to warehouse stock location. Transportation and distribution Management, 3(1), 27-31.

[8] Bartholdi, J.J., Hackman, S.T. (2011). Warehouse & distribution science, Release 0.94. Retrieved July, 28.

[9] Roodbergen, K. J., & Vis, I. F. (2009). A survey of literature on automated storage and retrieval systems. European journal of operational research, 194(2), 343-362.

[10] Dwivedi, V., Chauhan, T., Saxena, S., & Agrawal, P. (2012). Travelling salesman problem using genetic algorithm. IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI 2012), 1, 25.

[11] Silberholz, J., & Golden, B. (2007). The generalized traveling salesman problem: A new genetic algorithm approach. In extending the horizons: advances in computing, optimization, and decision technologies (pp. 165-181). Springer, Boston, MA.

[12] Banks, J., Carson, J., Nelson, B., & Nicol D. (2001). Discrete-Event System Simulation. Prentice Hall. p. 3. ISBN 0-13-088702-1.

[13] Chackelson, C., Errasti, A., Cipres, D., & Álvarez, M. J. (2011). Improving picking productivity by redesigning storage policy aided by simulations tools. In V international conference on industrial engineering and industrial management (pp. 306-313).

[14] Muppani, V. R., & Adil, G. K. (2008a). Efficient formation of storage classes for warehouse storage location assignment: a simulated annealing approach. Omega, 36(4), 609-618.

[15] Le-Duc, T., & De Koster, R. M. B. (2005). Travel distance estimation and storage zone optimization in a 2-block class-based storage strategy warehouse. International Journal of Production Research, 43(17), 3561-3581.

[16] Harmatuck DJ (1976). A comparison of two approaches to stock location. Logistics and Transportation Review; 12(4): 282–285.

[17] Graves, S.C., Hausman, W.H. and Schwarz, L.B. (1977). Storage-retrieval interleaving in automatic warehousing systems. Management Science 23, 935-945.

[18] Schwarz, L. B., Graves, S. C., & Hausman,W. H. (1978). Scheduling policies for automatic warehousing systems: Simulation results. AIIE Transactions, 10, 260–270.

[19] Roll, Y., & Rosenblatt, M. J. (1983). Random versus grouped storage policies and their effect on warehouse capacity. Material Flow, 1(3), 199-205.

[20] Rosenblatt, M.J., Roll, Y., 1984. Warehouse design with storage policy considerations. International Journal of Production Research 22 (5), 809–821.

[21] Frazele, E. A., & Sharp, G. P. (1989). Correlated assignment strategy can improve any order-picking operation. Industrial Engineering, 21(4), 33-37.

[22] Park, Y.H., Webster, D.B., (1989). Design of class-based storage racks for minimizing travel time in a three dimensional storage system. International Journal of Production Research 27 (9), 1589–1601.

[23] Malmborg, C. J., & Bhaskaran, K. (1990). A revised proof of optimality for the cube-per-order index rule for stored item location. Applied Mathematical Modelling, 14(2), 87-95.

[24] Gibson, D. R., & Sharp, G. P. (1992). Order batching procedures. European Journal of Operational Research, 58(1), 57-67.

[25] Van den Berg, J.P., (1996). Class-based storage allocation in a single command warehouse with space requirement constraints. International Journal of Industrial Engineering 3 (1), 21–28.

[26] Petersen, C. G., Aase, G. R., & Heiser, D. R. (2004). Improving order-picking performance through the implementation of class-based storage. International Journal of Physical Distribution & Logistics Management, 34(7), 534-544.

[27] Jane, C. C., & Laih, Y. W. (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. European Journal of Operational Research, 166(2), 489-496.

[28] Mantel, R. J., Schuur, P. C., & Heragu, S. S. (2007). Order oriented slotting: a new assignment strategy for warehouses. European Journal of Industrial Engineering, 1(3), 301-316.

[29] Muppani, V. R., & Adil, G. K. (2008b). A branch and bound algorithm for class based storage location assignment. European Journal of Operational Research, 189(2), 492-507.

[30] Bindi, F., Manzini, R., Pareschi, A., & Regattieri, A. (2009). Similarity-based storage allocation rules in an order picking system: an application to the food service industry. International Journal of Logistics Research and Applications, 12(4), 233-247.

[31] Xiao, J., & Zheng, L. (2010). A correlated storage location assignment problem in a single-block-multi-aisles warehouse considering BOM information. International Journal of Production Research, 48(5), 1321-1338.

[32] Accorsi, R., Manzini, R., & Bortolini, M. (2012). A hierarchical procedure for storage allocation and assignment within an order-picking system. A case study. International Journal of Logistics Research and Applications, 15(6), 351-364.

[33] Bottani, E., Cecconi, M., Vignali, G., & Montanari, R. (2012). Optimisation of storage allocation in order picking operations through a genetic algorithm. International Journal of Logistics Research and Applications, 15(2), 127-146.

[34] Ratliff, H. D., & Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Operations Research, 31(3), 507-521.

[35] Han, M.H., McGinnis, L.F., Shieh, J.S., White, J.A., (1987). On sequencing retrievals in an automated storage/retrieval system. IIE Transactions 19 (1), 56–66.

[36] Bachers, R., Dangelmaier, W., & Warnecke, H. J. (1988). Selection and use of order-picking strategies in a high-bay warehouse. Material flow, 4(4), 233-245.

[37] Bozer, Y.A., Schorn, E.C., Sharp, G.P., (1990). Geometric approaches to solve the Chebyshev traveling salesman problem. IIE Transactions 22 (3), 238–254.

[38] Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. IIE Transactions, 25(4), 77–87.

[39] Petersen, C. G., (1997). An evaluation of order picking routing policies. International Journal of Operations & Production Management, 17(11), 1098–1111.

[40] De Koster, R., & Van der Poort, E. (1998). Routing order pickers in a warehouse: A comparison between optimal and heuristic solutions. IIE Transactions, 30(5), 469–480.

[41] Roodbergen, K. J. (2001). Layout and routing methods for warehouses (No. EPS-2001-004-LIS).

[42] Hwang, H., Oh, Y. H., & Lee, Y. K. (2004). An evaluation of routing policies for order-picking operations in low-level picker-to-part system. International Journal of Production Research, 42(18), 3873-3889.

[43] Ene, S., & Öztürk, N. (2012). Storage location assignment and order picking optimization in the automotive industry. The international journal of advanced manufacturing technology, 60(5), 787-797.

[44] Mellema, P.M., Smith, C.A., (1988). Simulation analysis of narrow-aisle order selection systems. In: Proceedings of the 1988 Winter Simulation Conference, pp. 597–602.

[45] De Koster, R., (1994). Performance approximation of pick-to-belt order picking systems. European Journal of Operational Research 72, 558–573.

[46] Malmborg, C. J. (1995). Optimization of cubic-per-order index layout with zoning constraints. International Journal of Production Research, 33(2), 465–482.

[47] Jane, C.C., (2000). Storage location assignment in a distribution center. International Journal of Physical and Logistics Management 30 (1), 55–71.

[48] Petersen, C.G., (2002). Considerations in order picking zone configuration. International Journal of Operations & Production Management 27 (7), 793–805.

[49] Tsige, M. T. (2013). Improving order-picking efficiency via storage assignments strategies (Master's thesis, University of Twente).

[50] Elsayed, E.A., Unal, O.I., (1989). Order batching algorithms and travel-time estimation for automated storage/retrieval systems. International Journal of Production Research 27, 1097–1114.

[51] Pan, C.H., Liu, S.Y., (1995). A comparative study of order batching algorithms. Omega International Journal of Management Science 23(6), 691–700.

[52] Elsayed, E.A., Lee, M.K., Kim, S., Scherer, E., (1993). Sequencing and batching procedures for minimizing earliness and tardiness penalty of order retrievals. International Journal of Production Research 31 (3), 727–738.

[53] Elsayed, E.A., Lee, M.K., (1996). Order processing in automated storage/retrieval systems with due dates. IIE Transactions 28, 567– 577.

[54] Ruben, R. A., & Jacobs, F. R. (1999). Batch construction heuristics and storage assignment strategies for walk/ride and pick systems. Management Science, 45, 575–596.

[55] Gademann, N., Van de Velde, S., (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions 37 (1), 63–75.

[56] Won, J., Olafsson, S., (2005). Joint order batching and order picking in warehouse operations. International Journal of Production Research 43 (7), 1427–1442.

[57] Lee, M. K. (1992). A storage assignment policy in a man-on-board automated storage/retrieval system. The International Journal of Production Research, 30(10), 2281-2292.

[58] Malmborg, C. J., & Al-Tassan, K. (2000). An integrated performance model for order picking systems with randomized storage. Applied Mathematical Modelling, 24(2), 95-111.

[59] Sultana, M.F. (2008). Optimal planning of store allocation and assignment of spare parts in a warehouse (Unpublished Master’s thesis, Khulna University of Engineering & Technology).

[60] Le-Duc, T. (2005). Design and control of efficient order picking processes.

[61] Islam, S. A. (2011). Development of storage allocation models in a warehouse (Unpublished Master’s thesis, Khulna University of Engineering & Technology).

[62] Wang, S., Lu, Z., Wei, L., Ji, G., & Yang, J. (2016). Fitness-scaling adaptive genetic algorithm with local search for solving the Multiple Depot Vehicle Routing Problem. Simulation, 92(7), 601-616.

[63] Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, 13(2), 129-170.

APPENDIX

The code for the simulation program:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

FILE *fp;

void main() {
       srand(time(NULL));
       //FILE *fp;
       fp=fopen("output.txt","w");
int cs[43][43];
int cf[43][43];
int cg[43][43];
int cc[43][43];
int m1[6]={15,7,42,28,36,6}, m2[7]={9,24,18,16,22,14,32}, m3[5]={25,41,21,34,5},
m4[6]={11,1,35,26,13,31},m5[8]={3,4,40,20,17,10,2,8},m6[4]={33,19,12,29},
m7[6]={23,38,30,37,39,27};

 float R, p[7]={0.130,0.321,0.450,0.599,0.753,0.814,1}, p1[6]={0.265,0.505,0.699,0.852,0.959,1},
             p2[7]={0.216,0.415,0.603,0.777,0.902,0.968,1}, p3[5]={0.335,0.577,0.742,0.881,1},
p4[6]={0.259,0.496,0.679,0.822,0.938,1}, p5[8]={0.255,0.411,0.558,0.666,0.766,0.852,0.930,1},
 p6[4]={0.326,0.609,0.837,1},p7[6]={0.257,0.464,0.653,0.799,0.906,1};

   int a[10],n,i,j,k,l,s,order_size;
   float cost,tr_cost=0,ts_cost=0,tm_cost=0,tc_cost=0,avg;

   printf("Enter the total number of maintenance:");
   scanf("%d",&n);

   fprintf(fp,"\nNO.  Type   Total Parts   Parts No. Sequence based   Distance     Avg. Distance
Frequency based   Distance    Avg. Distance    Group and frequency based    Distance     Avg. Distance
Group and COI based   Distance    Avg. Distance\n" );

   for(i=0;i<n;i++){
          R=(float)rand()/(float)(RAND_MAX+1);
          for(j=0;j<7;j++){
                 if(R<=p[j]){
                     s=j;
                     break;
                      }
          }

   a[0]=0;
          if(s==0){
             order_size=rand()%6;
             if(order_size==0)
                     order_size=6;

               for(k=1;k<=order_size;k++){
                   R=(float)rand()/(float)(RAND_MAX+1);

                   for(j=0;j<6;j++){
                          if(R<=p1[j]){
                                 for(l=1;l<k;l++){
```

```
                                if(a[l]==m1[j]){
                                R=(float)rand()/(float)(RAND_MAX+1);
                                j=-1;
                                break;}
                                      }
                             if(l==k){
                               a[k]=m1[j];
                               break;
                                   }
                               }
                           }
                      }
             }

      else if(s==1){
         order_size=rand()%7;
         if(order_size==0)
                 order_size=7;

           for(k=1;k<=order_size;k++){
                 R=(float)rand()/(float)(RAND_MAX+1);

                 for(j=0;j<7;j++){
                       if(R<=p2[j]){
                             for(l=1;l<k;l++){
                                if(a[l]==m2[j]){
                                R=(float)rand()/(float)(RAND_MAX+1);
                                j=-1;
                                break;}
                                      }
                             if(l==k){
                               a[k]=m2[j];
                               break;
                                   }
                               }
                      }
                 }
             }

      else if(s==2){
         order_size=rand()%5;
         if(order_size==0)
                 order_size=5;

           for(k=1;k<=order_size;k++){
                 R=(float)rand()/(float)(RAND_MAX+1);

                 for(j=0;j<5;j++){
                       if(R<=p3[j]){
                             for(l=1;l<k;l++){
                                if(a[l]==m3[j]){
                                R=(float)rand()/(float)(RAND_MAX+1);
                                j=-1;
                                break;}
                                      }
                             if(l==k){
                               a[k]=m3[j];
                               break;
                                   }
                               }
```

```
                    }
                }
            }


else if(s==3){
   order_size=rand()%6;
   if(order_size==0)
            order_size=6;

     for(k=1;k<=order_size;k++){
          R=(float)rand()/(float)(RAND_MAX+1);

          for(j=0;j<6;j++){
                 if(R<=p4[j]){
                        for(l=1;l<k;l++){
                             if(a[l]==m4[j]){
                             R=(float)rand()/(float)(RAND_MAX+1);
                             j=-1;
                             break;}
                                      }
                        if(l==k){
                           a[k]=m4[j];
                           break;
                               }
                           }
                       }
                }
            }

else if(s==4){
   order_size=rand()%8;
   if(order_size==0)
            order_size=8;

     for(k=1;k<=order_size;k++){
          R=(float)rand()/(float)(RAND_MAX+1);

          for(j=0;j<8;j++){
                 if(R<=p5[j]){
                        for(l=1;l<k;l++){
                             if(a[l]==m5[j]){
                             R=(float)rand()/(float)(RAND_MAX+1);
                             j=-1;
                             break;}
                                      }
                        if(l==k){
                           a[k]=m5[j];
                           break;
                               }
                           }
                       }
                }
            }

else if(s==5){
   order_size=rand()%4;
   if(order_size==0)
            order_size=4;
```

```
              for(k=1;k<=order_size;k++){
                    R=(float)rand()/(float)(RAND_MAX+1);

                    for(j=0;j<4;j++){
                            if(R<=p6[j]){
                                    for(l=1;l<k;l++){
                                         if(a[l]==m6[j]){
                                         R=(float)rand()/(float)(RAND_MAX+1);
                                         j=-1;
                                         break;}
                                                    }
                                    if(l==k){
                                       a[k]=m6[j];
                                       break;
                                           }
                                       }
                                   }
                           }
                    }

        else{
            order_size=rand()%6;
            if(order_size==0)
                      order_size=6;

              for(k=1;k<=order_size;k++){
                    R=(float)rand()/(float)(RAND_MAX+1);

                    for(j=0;j<6;j++){
                            if(R<=p7[j]){
                                    for(l=1;l<k;l++){
                                         if(a[l]==m7[j]){
                                         R=(float)rand()/(float)(RAND_MAX+1);
                                         j=-1;
                                         break;}
                                                    }
                                    if(l==k){
                                       a[k]=m7[j];
                                       break;
                                           }
                                       }
                                   }
                           }
                    }
a[k]=0;
printf("\n%d\t%d\t%d\t",i+1,s+1,order_size);
fprintf(fp,"\n%d\t%d\t%d\t",i+1,s+1,order_size);
for(j=1;j<order_size+1;j++){
            printf("%3d",a[j]);
            fprintf(fp,"%3d",a[j]);
                     }

 cost=TSP(order_size+2,a,cs);
    tr_cost=tr_cost+cost;
    avg=tr_cost/(i+1);
         printf("\t%.1f",avg);
         fprintf(fp,"\t%.1f",avg);
 cost=TSP(order_size+2,a,cf);
    ts_cost=ts_cost+cost;
    avg=ts_cost/(i+1);
```

```
            printf("\t%.1f",avg);
           fprintf(fp,"\t%.1f",avg);
   cost=TSP(order_size+2,a,cg);
       tm_cost=tm_cost+cost;
       avg=tm_cost/(i+1);
            printf("\t%.1f",avg);
           fprintf(fp,"\t%.1f",avg);
   cost=TSP(order_size+2,a,cc);
       tc_cost=tc_cost+cost;
       avg=tc_cost/(i+1);
            printf("\t%.1f",avg);
           fprintf(fp,"\t%.1f",avg);
                          }  //END for Maintenance type
      printf("\n\n\nTotal Distance for Sequence based arrangement= %.0f",tr_cost);
      printf("\nTotal Distance for Frequency based arrangement= %.0f",ts_cost);
      printf("\nTotal Distance for Group and frequency based arrangement= %.0f",tm_cost);
      printf("\nTotal Distance for Group and COI based arrangement= %.0f",tc_cost);
      fprintf(fp,"\n\n\nTotal Distance for Sequence based arrangement= %.0f",tr_cost);
      fprintf(fp,"\nTotal Distance for Frequency based arrangement= %.0f",ts_cost);
      fprintf(fp,"\nTotal Distance for Group and frequency based arrangement= %.0f",tm_cost);
      fprintf(fp,"\nTotal Distance for Group and COI based arrangement= %.0f",tc_cost);
   }


////////////////////////////////////////////////////////////////////////////////GENETIC ALGORITHM FUNCTION

float TSP(int n, int a[], int c[43][43]) {

int p[100][10],new_p[100][10],c1[10],c2[10],x=0,p_size,i,j,k;
int g,generation,sorted_p[100],temp,p1,p2,e,new_p_size,s1,s2,r1,r2,check;
float cost[100],t_cost[100],total,t_temp,q[100],R,xyz;
int digit,t_digit;

/////////////////////////////////////////////Population Size
   if(n==10)
      p_size=56;
   else if(n==9)
      p_size=42;
   else if(n==8)
      p_size=30;
   else if(n==7)
      p_size=24;
   else if(n==6)
      p_size=12;
   else
      p_size=6;

//////////////////////////////////////////////Elite Size
   if(n==10)
      e=12;
   else if(n==9)
      e=10;
   else if(n==8)
      e=8;
   else if(n==7)
      e=6;
   else if(n==6)
      e=4;
   else
      e=2;
```

```
//////////////////////////////////////////////////Initial Population Creation
   if(n==3){
      if(c[1][2]==10){
      i=a[1];
      printf("   %3d",i);
      printf("  %4d",c[0][i]+c[i][0]);
      fprintf(fp,"\t\t\t%3d",i);
      fprintf(fp,"\t\t\t%4d",c[0][i]+c[i][0]);
      xyz=c[0][i]+c[i][0];
      return xyz;
        }
      else{
      i=a[1];
      printf("   %3d",i);
      printf("  %4d",c[0][i]+c[i][0]);
      fprintf(fp,"\t\t%3d",i);
      fprintf(fp,"\t\t\t%4d",c[0][i]+c[i][0]);
      xyz=c[0][i]+c[i][0];
      return xyz;
        }
       }

   else if(n==4){
      if(c[1][2]==10){
      i=a[1];
      j=a[2];
      r1=c[0][i]+c[i][j]+c[j][0];
      r2=c[0][j]+c[j][i]+c[i][0];
      if(r1<r2){
      printf("            %3d%3d",i,j);
      printf("          %4d",r1);
      fprintf(fp,"\t\t\t%3d%3d",i,j);
      fprintf(fp,"\t\t\t%4d",r1);
      xyz=r1;
      return xyz;
      }
      else{
      printf("            %3d%3d",j,i);
      printf("          %4d",r2);
      fprintf(fp,"\t\t\t%3d%3d",j,i);
      fprintf(fp,"\t\t\t%4d",r2);
      xyz=r2;
      return xyz;
       }
        }
     else{
      i=a[1];
      j=a[2];
      r1=c[0][i]+c[i][j]+c[j][0];
      r2=c[0][j]+c[j][i]+c[i][0];
      if(r1<r2){
      printf("            %3d%3d",i,j);
      printf("          %4d",r1);
      fprintf(fp,"\t\t%3d%3d",i,j);
      fprintf(fp,"\t\t\t%4d",r1);
      xyz=r1;
      return xyz;
      }
      else{
      printf("            %3d%3d",j,i);
      printf("          %4d",r2);
      fprintf(fp,"\t\t%3d%3d",j,i);
      fprintf(fp,"\t\t\t%4d",r2);
      xyz=r2;
      return xyz;
        }
      }
             }
   else{
   for(i=0;i<n;i++){
     for(j=1;j<n-2;j++){
         for(k=0;k<n;k++)
              p[x][k]=a[k];
         x++;
         if(x==p_size)
            break;
         temp=a[j];
         a[j]=a[j+1];
         a[j+1]=temp;
              }
      if(x==p_size)
         break;
       }

////////////////////////////////////////////////////////Fitness Value Calculation

  for(g=1;g<=100;g++){
    total=0;//For Roulette Wheel
    for(i=0;i<p_size;i++){
           cost[i]=0;
        for(j=0;j<n-1;j++){
              p1=p[i][j];
              p2=p[i][j+1];
              cost[i]=cost[i]+c[p1][p2];
                       }
        cost[i]=1/cost[i];
        total=total+cost[i];
                }
    for(i=0;i<p_size;i++)
           t_cost[i]=cost[i];

/////////////////////////////////////////////Sorting fitness value in t_cost[]

    for(i=0;i<p_size;i++){
      for(j=0;j<x-1-i;j++){
         if(t_cost[j]<t_cost[j+1]){
              t_temp=t_cost[j];
              t_cost[j]=t_cost[j+1];
              t_cost[j+1]=t_temp;
                     }
                  }
              }

/////////////////////////////////Sorting Population number according to fitness value

    for(i=0;i<p_size;i++){
        for(j=0;j<p_size;j++){
               if(t_cost[i]==cost[j]){
                     for(k=0;k<i;k++){
                          if(j==sorted_p[k])
```

```
                                        break;
                                          }
                          if(k==i){
                               sorted_p[i]=j;
                               break;
                                   }
                               }
                             }
                         }

//////////////////////////////////////////////Roulette Wheel Point Creation in q[i]

    j=sorted_p[0];
    //printf("\n\n\nBest Result for Generation-%d: %.0f",g,1/cost[j]);
    //printf("\nSequence: ");
    //fprintf(fp,"\n\n\nBest Result for Generation-%d: %.0f",g,1/cost[j]);
    //fprintf(fp,"\nSequence: ");
    if(g==100){
      if(n<6){
         if(c[1][2]==10){
        printf("           ");
        fprintf(fp,"\t\t\t");
        for(i=1;i<n-1;i++){
            printf("%3d",p[j][i]);
            fprintf(fp,"%3d",p[j][i]);
                   }
      printf("        %.0f",1/cost[j]);
      fprintf(fp,"\t\t\t%.0f",1/cost[j]);
                    }
         else{
        printf("           ");
        fprintf(fp,"\t\t");
        for(i=1;i<n-1;i++){
            printf("%3d",p[j][i]);
            fprintf(fp,"%3d",p[j][i]);
                   }

      printf("        %.0f",1/cost[j]);
      fprintf(fp,"\t\t\t%.0f",1/cost[j]);

         }
      }

     else  if(n==6){
                t_digit=0;
             for(i=1;i<=4;i++){
                     if(p[j][i]<10)
                        digit=1;
                     else
                        digit=2;
                     t_digit=t_digit+digit;
                          }
                 if(t_digit<6){
                      if(c[1][2]==10){
                     printf("           ");
                     fprintf(fp,"\t\t\t");
                     for(i=1;i<n-1;i++){
                     printf("%3d",p[j][i]);
                     fprintf(fp,"%3d",p[j][i]);
                           }
                   printf("         %.0f",1/cost[j]);
                   fprintf(fp,"\t\t\t%.0f",1/cost[j]);
                                  }
                         else{
                     printf("            ");
                     fprintf(fp,"\t\t");
                     for(i=1;i<n-1;i++){
                     printf("%3d",p[j][i]);
                     fprintf(fp,"%3d",p[j][i]);
                           }

                   printf("         %.0f",1/cost[j]);
                   fprintf(fp,"\t\t\t%.0f",1/cost[j]);
                         }
                           }
                   else {
                     printf("            ");
                     fprintf(fp,"\t\t");
                     for(i=1;i<n-1;i++){
                     printf("%3d",p[j][i]);
                     fprintf(fp,"%3d",p[j][i]);
                           }

                   printf("         %.0f",1/cost[j]);
                   fprintf(fp,"\t\t%.0f",1/cost[j]);
                    }
                    }
         else if(n<10){
        printf("            ");
        fprintf(fp,"\t\t");
        for(i=1;i<n-1;i++){
            printf("%3d",p[j][i]);
            fprintf(fp,"%3d",p[j][i]);
                   }

      printf("         %.0f",1/cost[j]);
      fprintf(fp,"\t\t%.0f",1/cost[j]);
               }
     else{
         if(c[1][2]==10){
         printf("            ");
         fprintf(fp,"\t");
         for(i=1;i<n-1;i++){
            printf("%3d",p[j][i]);
            fprintf(fp,"%3d",p[j][i]);
                   }
            //fprintf(fp,"%3d",p[j][i]);
        printf("         %.0f",1/cost[j]);
        fprintf(fp,"\t%.0f",1/cost[j]);
            }
         else {
         printf("            ");
         fprintf(fp,"\t\t");
         for(i=1;i<n-1;i++){
            printf("%3d",p[j][i]);
            fprintf(fp,"%3d",p[j][i]);
                   }
            //fprintf(fp,"%3d",p[j][i]);
        printf("         %.0f",1/cost[j]);
```

```
        fprintf(fp,"\t%.0f",1/cost[j]);
         }
              }
        xyz=1/cost[j];
        return xyz;
               }
    q[0]=cost[j]/total;

    for(i=1;i<p_size;i++){
         j=sorted_p[i];
         q[i]=q[i-1]+(cost[j]/total);
              }

///////////////////////////////////////////////////New Chromosomes Creation in new_p[][]

////////////////////////////Elite Chromosomes Selection

    for(i=0;i<e;i++){
         j=sorted_p[i];
         for(k=0;k<n;k++)
                new_p[i][k]=p[j][k];
                  }

/////////////////////////////////////////////Remaining Chromosomes Creation by Cross Over

    new_p_size=e;
    while(1){
          if(new_p_size==p_size){
              for(i=0;i<p_size;i++){
                    for(j=0;j<n;j++)
                          p[i][j]=new_p[i][j];
                                  }
              break;
                           }
          ////////////////////////////////////Selection of two parents
          R=(float)rand()/(float)(RAND_MAX+1);
          for(i=0;i<p_size;i++){
                    if(R<=q[i]){
                         s1=i;
                         break;
                           }
                     }

          R=(float)rand()/(float)(RAND_MAX+1);
          for(i=0;i<p_size;i++){
                    if(R<=q[i]){
                         s2=i;
                         if(s1==s2){
                              R=(float)rand()/(float)(RAND_MAX+1);
                              i=0;
                              }
                         else
                         break;
                           }
                     }


////////////////////////////////////////CROSS OVER Between s1 and s2 parents
          r1=rand()%n;
          r2=rand()%n;
```

```
           while(r1>=r2 || r1==0 || r2==n-1){
               r1=rand()%n;
               r2=rand()%n;
                   }
           for(i=r1;i<=r2;i++){
                c1[i]=p[s2][i];
                c2[i]=p[s1][i];
                       }
//////////////////////////////////////////////////////////////Child 1 Creation

           j=0;
           for(i=0;i<n;i++){
                   if(i>=r1 && i<=r2)
                         continue;
                   while(1){
                        check=0;
                        for(k=r1;k<=r2;k++){
                              if(p[s1][j]==c1[k]){
                                      j++;
                                      check++;
                                      break;
                                             }
                                          }
                        if(check==0){
                            c1[i]=p[s1][j];
                            j++;
                            break;
                             }
                          }
                     }
//////////////////////////////////////////////////////////////MUTATION of Child 1

          R=(float)rand()/(float)(RAND_MAX+1);
          if(R<.01){
             r1=rand()%n;
             r2=rand()%n;
             while(r1>=r2 || r1==0 || r2==n-1){
                r1=rand()%n;
                r2=rand()%n;
                    }
             k=c1[r1];
             c1[r1]=c1[r2];
             c1[r2]=k;
             }

          for(i=0;i<n;i++)
              new_p[new_p_size][i]=c1[i];
          new_p_size++;

//////////////////////////////////////////////////////////////Child 2 Creation

           j=0;
           for(i=0;i<n;i++){
                   if(i>=r1 && i<=r2)
                         continue;
                   while(1){
                        check=0;
                        for(k=r1;k<=r2;k++){
                              if(p[s2][j]==c2[k]){
                                      j++;
```

```
                                    check++;
                                    break;
                                         }
                                       }
                            if(check==0){
                               c2[i]=p[s2][j];
                               j++;
                               break;
                                }
                              }
                          }

////////////////////////////////////////////////////////////MUTATION of Child 2
         R=(float)rand()/(float)(RAND_MAX+1);
         if(R<.01){
           r1=rand()%n;
           r2=rand()%n;
           while(r1>=r2 || r1==0 || r2==n-1){
              r1=rand()%n;
              r2=rand()%n;
                  }
           k=c2[r1];
           c2[r1]=c2[r2];
           c2[r2]=k;
           }
        for(i=0;i<n;i++)
            new_p[new_p_size][i]=c1[i];
        new_p_size++;

             }
        }
    }
 }
```

Distance Matrix for Sequence Based Storage

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**
0	0	15	13	11	9	7	5	3	17	15	13	11	9	7	5	19	17	15	13	11	9	7	21	19	17	15	13	11	9	23	21	19	17	15	13	11	25	23	21	19	17	15	13
1	15	0	2	4	6	8	10	12	2	4	6	8	10	12	14	4	6	8	10	12	14	16	6	8	10	12	14	16	18	8	10	12	14	16	18	20	10	12	14	16	18	20	22
2	13	2	0	2	4	6	8	10	4	2	4	6	8	10	12	6	4	6	8	10	12	14	8	6	8	10	12	14	16	10	8	10	12	14	16	18	12	10	12	14	16	18	20
3	11	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12	10	8	6	8	10	12	14	12	10	8	10	12	14	16	14	12	10	12	14	16	18
4	9	6	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12	10	8	6	8	10	12	14	12	10	8	10	12	14	16	14	12	10	12	14	16
5	7	8	6	4	2	0	2	4	10	8	6	4	2	4	6	12	10	8	6	4	6	8	14	12	10	8	6	8	10	16	14	12	10	8	10	12	18	16	14	12	10	12	14
6	5	10	8	6	4	2	0	2	12	10	8	6	4	2	4	14	12	10	8	6	4	6	16	14	12	10	8	6	8	18	16	14	12	10	8	10	20	18	16	14	12	10	12
7	3	12	10	8	6	4	2	0	14	12	10	8	6	4	2	16	14	12	10	8	6	4	18	16	14	12	10	8	6	20	18	16	14	12	10	8	22	20	18	16	14	12	10
8	17	2	4	6	8	10	12	14	0	2	4	6	8	10	12	2	4	6	8	10	12	14	4	6	8	10	12	14	16	6	8	10	12	14	16	18	8	10	12	14	16	18	20
9	15	4	2	4	6	8	10	12	2	0	2	4	6	8	10	4	2	4	6	8	10	12	6	4	6	8	10	12	14	8	6	8	10	12	14	16	10	8	10	12	14	16	18
10	13	6	4	2	4	6	8	10	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12	10	8	6	8	10	12	14	12	10	8	10	12	14	16
11	11	8	6	4	2	4	6	8	6	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12	10	8	6	8	10	12	14	12	10	8	10	12	14
12	9	10	8	6	4	2	4	6	8	6	4	2	0	2	4	10	8	6	4	2	4	6	12	10	8	6	4	6	8	14	12	10	8	6	8	10	16	14	12	10	8	10	12
13	7	12	10	8	6	4	2	4	10	8	6	4	2	0	2	12	10	8	6	4	2	4	14	12	10	8	6	4	6	16	14	12	10	8	6	8	18	16	14	12	10	8	10
14	5	14	12	10	8	6	4	2	12	10	8	6	4	2	0	14	12	10	8	6	4	2	16	14	12	10	8	6	4	18	16	14	12	10	8	6	20	18	16	14	12	10	8
15	19	4	6	8	10	12	14	16	2	4	6	8	10	12	14	0	2	4	6	8	10	12	2	4	6	8	10	12	14	4	6	8	10	12	14	16	6	8	10	12	14	16	18
16	17	6	4	6	8	10	12	14	4	2	4	6	8	10	12	2	0	2	4	6	8	10	4	2	4	6	8	10	12	6	4	6	8	10	12	14	8	6	8	10	12	14	16
17	15	8	6	4	6	8	10	12	6	4	2	4	6	8	10	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12	10	8	6	8	10	12	14
18	13	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12	10	8	6	8	10	12
19	11	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	10	8	6	4	2	4	6	12	10	8	6	4	6	8	14	12	10	8	6	8	10
20	9	14	12	10	8	6	4	6	12	10	8	6	4	2	4	10	8	6	4	2	0	2	12	10	8	6	4	2	4	14	12	10	8	6	4	6	16	14	12	10	8	6	8
21	7	16	14	12	10	8	6	4	14	12	10	8	6	4	2	12	10	8	6	4	2	0	14	12	10	8	6	4	2	16	14	12	10	8	6	4	18	16	14	12	10	8	6
22	21	6	8	10	12	14	16	18	4	6	8	10	12	14	16	2	4	6	8	10	12	14	0	2	4	6	8	10	12	2	4	6	8	10	12	14	4	6	8	10	12	14	16
23	19	8	6	8	10	12	14	16	6	4	6	8	10	12	14	4	2	4	6	8	10	12	2	0	2	4	6	8	10	4	2	4	6	8	10	12	6	4	6	8	10	12	14
24	17	10	8	6	8	10	12	14	8	6	4	6	8	10	12	6	4	2	4	6	8	10	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10	12
25	15	12	10	8	6	8	10	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	6	8	6	4	2	4	6	8	10	8	6	4	6	8	10
26	13	14	12	10	8	6	8	10	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	10	8	6	4	2	4	6	12	10	8	6	4	6	8
27	11	16	14	12	10	8	6	8	14	12	10	8	6	4	6	12	10	8	6	4	2	4	10	8	6	4	2	0	2	12	10	8	6	4	2	4	14	12	10	8	6	4	6
28	9	18	16	14	12	10	8	6	16	14	12	10	8	6	4	14	12	10	8	6	4	2	12	10	8	6	4	2	0	14	12	10	8	6	4	2	16	14	12	10	8	6	4
29	23	8	10	12	14	16	18	20	6	8	10	12	14	16	18	4	6	8	10	12	14	16	2	4	6	8	10	12	14	0	2	4	6	8	10	12	2	4	6	8	10	12	14
30	21	10	8	10	12	14	16	18	8	6	8	10	12	14	16	6	4	6	8	10	12	14	4	2	4	6	8	10	12	2	0	2	4	6	8	10	4	2	4	6	8	10	12
31	19	12	10	8	10	12	14	16	10	8	6	8	10	12	14	8	6	4	6	8	10	12	6	4	2	4	6	8	10	4	2	0	2	4	6	8	6	4	2	4	6	8	10
32	17	14	12	10	8	10	12	14	12	10	8	6	8	10	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	6	8	6	4	2	4	6	8
33	15	16	14	12	10	8	10	12	14	12	10	8	6	8	10	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	10	8	6	4	2	4	6
34	13	18	16	14	12	10	8	10	16	14	12	10	8	6	8	14	12	10	8	6	4	6	12	10	8	6	4	2	4	10	8	6	4	2	0	2	12	10	8	6	4	2	4
35	11	20	18	16	14	12	10	8	18	16	14	12	10	8	6	16	14	12	10	8	6	4	14	12	10	8	6	4	2	12	10	8	6	4	2	0	14	12	10	8	6	4	2
36	25	10	12	14	16	18	20	22	8	10	12	14	16	18	20	6	8	10	12	14	16	18	4	6	8	10	12	14	16	2	4	6	8	10	12	14	0	2	4	6	8	10	12
37	23	12	10	12	14	16	18	20	10	8	10	12	14	16	18	8	6	8	10	12	14	16	6	4	6	8	10	12	14	4	2	4	6	8	10	12	2	0	2	4	6	8	10
38	21	14	12	10	12	14	16	18	12	10	8	10	12	14	16	10	8	6	8	10	12	14	8	6	4	6	8	10	12	6	4	2	4	6	8	10	4	2	0	2	4	6	8
39	19	16	14	12	10	12	14	16	14	12	10	8	10	12	14	12	10	8	6	8	10	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4	6
40	17	18	16	14	12	10	12	14	16	14	12	10	8	10	12	14	12	10	8	6	8	10	12	10	8	6	4	6	8	10	8	6	4	2	4	6	8	6	4	2	0	2	4
41	15	20	18	16	14	12	10	12	18	16	14	12	10	8	10	16	14	12	10	8	6	8	14	12	10	8	6	4	6	12	10	8	6	4	2	4	10	8	6	4	2	0	2
42	13	22	20	18	16	14	12	10	20	18	16	14	12	10	8	18	16	14	12	10	8	6	16	14	12	10	8	6	4	14	12	10	8	6	4	2	12	10	8	6	4	2	0

Distance Matrix for Frequency Based Storage

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**
0	0	7	19	9	11	19	21	11	23	5	21	9	15	19	17	11	11	19	7	15	15	15	13	3	7	5	13	17	15	25	9	23	21	13	17	13	17	15	9	17	13	11	13
1	7	0	12	6	4	12	14	12	16	6	14	10	8	12	10	8	4	12	4	16	8	8	6	4	8	2	6	14	8	18	2	16	14	10	10	6	10	12	2	10	14	4	6
2	19	12	0	10	8	8	2	8	4	14	6	10	8	4	2	8	8	4	12	12	4	4	6	16	12	14	6	10	4	6	10	4	2	6	6	10	2	8	10	6	10	8	6
3	9	6	10	0	10	10	12	6	14	4	12	4	10	10	8	2	2	10	2	10	6	6	4	6	2	4	8	8	6	16	8	14	12	4	8	12	8	6	4	8	8	6	4
4	11	4	8	10	0	16	10	16	12	10	14	14	4	12	6	12	8	8	8	20	4	8	6	8	12	6	2	18	12	14	2	12	10	14	6	2	10	16	6	14	18	4	10
5	19	12	8	10	16	0	6	8	8	14	2	10	16	4	10	8	8	12	12	4	12	8	10	16	12	14	14	2	4	6	14	4	10	6	14	18	6	4	10	2	6	12	6
6	21	14	2	12	10	6	0	10	2	16	4	12	10	2	4	10	10	6	14	10	6	6	8	18	14	16	8	8	6	4	12	2	4	8	8	12	4	6	12	4	8	10	8
7	11	12	8	6	16	8	10	0	12	6	10	2	16	8	10	4	8	12	8	4	12	8	10	8	4	10	14	6	4	14	14	12	10	2	14	18	6	4	10	6	2	12	6
8	23	16	4	14	12	8	2	12	0	18	6	14	8	4	6	12	12	4	16	12	8	8	10	20	16	18	10	10	8	2	14	4	2	10	6	10	6	8	14	6	10	12	10
9	5	6	14	4	10	14	16	6	18	0	16	4	10	14	12	6	6	14	2	10	10	10	8	2	2	4	8	12	10	20	8	18	16	8	12	12	12	10	4	12	8	6	8
10	21	14	6	12	14	2	4	10	6	16	0	12	14	2	8	10	10	10	14	6	10	6	8	18	14	16	12	4	6	4	12	2	8	8	12	16	4	6	12	4	8	10	8
11	9	10	10	4	14	10	12	2	14	4	12	0	14	10	8	2	6	10	6	6	10	6	8	6	2	8	12	8	6	16	12	14	12	4	12	16	8	6	8	8	4	10	4
12	15	8	8	10	4	16	10	16	8	10	14	14	0	12	6	12	8	4	8	20	4	8	6	12	12	10	2	18	12	10	6	12	6	14	2	2	10	16	6	14	18	4	10
13	19	12	4	10	12	4	2	8	4	14	2	10	12	0	6	8	8	8	12	8	8	4	6	16	12	14	10	6	4	6	10	4	6	6	10	14	2	4	10	2	6	8	6
14	17	10	2	8	6	10	4	10	6	12	8	8	6	6	0	6	6	2	10	14	2	2	4	14	10	12	4	12	6	8	8	6	4	8	4	8	4	10	8	8	12	6	4
15	11	8	8	2	12	8	10	4	12	6	10	2	12	8	6	0	4	8	4	8	8	4	6	8	4	6	10	6	4	14	10	12	10	2	10	14	6	4	6	6	6	8	2
16	11	4	8	2	8	8	10	8	12	6	10	6	8	8	6	4	0	8	4	12	4	4	2	8	4	6	6	10	4	14	6	12	10	6	6	10	6	8	2	6	10	4	2
17	19	12	4	10	8	12	6	12	4	14	10	10	4	8	2	8	8	0	12	16	4	4	6	16	12	14	6	14	8	6	10	8	2	10	2	6	6	12	10	10	14	8	6
18	7	4	12	2	8	12	14	8	16	2	14	6	8	12	10	4	4	12	0	12	8	8	6	4	4	2	6	10	8	18	6	16	14	6	10	10	10	8	2	10	10	4	6
19	15	16	12	10	20	4	10	4	12	10	6	6	20	8	14	8	12	16	12	0	16	12	14	12	8	14	18	2	8	10	18	8	14	6	18	22	10	4	14	6	2	16	10
20	15	8	4	6	4	12	6	12	8	10	10	10	4	8	2	8	4	4	8	16	0	4	2	12	8	10	2	14	8	10	6	8	6	10	2	6	6	12	6	10	14	4	6
21	15	8	4	6	8	8	6	8	8	10	6	6	8	4	2	4	4	4	8	12	4	0	2	12	8	10	6	10	4	10	6	8	6	6	6	10	2	8	6	6	10	4	2
22	13	6	6	4	6	10	8	10	10	8	8	8	6	6	4	6	2	6	6	14	2	2	0	10	6	8	4	12	6	12	4	10	8	8	4	8	4	10	4	8	12	2	4
23	3	4	16	6	8	16	18	8	20	2	18	6	12	16	14	8	8	16	4	12	12	12	10	0	4	2	10	14	12	22	6	20	18	10	14	10	14	12	6	14	10	8	10
24	7	8	12	2	12	12	14	4	16	2	14	2	12	12	10	4	4	12	4	8	8	8	6	4	0	6	10	10	8	18	10	16	14	6	10	14	10	8	6	10	6	8	6
25	5	2	14	4	6	14	16	10	18	4	16	8	10	14	12	6	6	14	2	14	10	10	8	2	6	0	8	12	10	20	4	18	16	8	12	8	12	10	4	12	12	6	8
26	13	6	6	8	2	14	8	14	10	8	12	12	2	10	4	10	6	6	6	18	2	6	4	10	10	8	0	16	10	12	4	10	8	12	4	4	8	14	4	12	16	2	8
27	17	14	10	8	18	2	8	6	10	12	4	8	18	6	12	6	10	14	10	2	14	10	12	14	10	12	16	0	6	8	16	6	12	4	16	20	8	2	12	4	4	14	8
28	15	8	4	6	12	4	6	4	8	10	6	6	12	4	6	4	4	8	8	8	8	4	6	12	8	10	10	6	0	10	10	8	6	2	10	14	2	4	6	2	6	8	2
29	25	18	6	16	14	6	4	14	2	20	4	16	10	6	8	14	14	6	18	10	10	10	12	22	18	20	12	8	10	0	16	2	4	12	8	12	8	10	16	8	12	14	12
30	9	2	10	8	2	14	12	14	14	8	12	12	6	10	8	10	6	10	6	18	6	6	4	6	10	4	4	16	10	16	0	14	12	12	8	4	8	14	4	12	16	2	8
31	23	16	4	14	12	4	2	12	4	18	2	14	12	4	6	12	12	8	16	8	8	8	10	20	16	18	10	6	8	2	14	0	6	10	10	14	6	8	14	6	10	12	10
32	21	14	2	12	10	10	4	10	2	16	8	12	6	6	4	10	10	2	14	14	6	6	8	18	14	16	8	12	6	4	12	6	0	8	4	8	4	10	12	8	12	10	8
33	13	10	6	4	14	6	8	2	10	8	8	4	14	6	8	2	6	10	6	6	10	6	8	10	6	8	12	4	2	12	12	10	8	0	12	16	4	2	8	4	4	10	4
34	17	10	6	8	6	14	8	14	6	12	12	12	2	10	4	10	6	2	10	18	2	6	4	14	10	12	4	16	10	8	8	10	4	12	0	4	8	14	8	12	16	6	8
35	13	6	10	12	2	18	12	18	10	12	16	16	2	14	8	14	10	6	10	22	6	10	8	10	14	8	4	20	14	12	4	14	8	16	4	0	12	18	8	16	20	6	12
36	17	10	2	8	10	6	4	6	6	12	4	8	10	2	4	6	6	6	10	10	6	2	4	14	10	12	8	8	2	8	8	6	4	4	8	12	0	6	8	4	8	6	4
37	15	12	8	6	16	4	6	4	8	10	6	6	16	4	10	4	8	12	8	4	12	8	10	12	8	10	14	2	4	10	14	8	10	2	14	18	6	0	10	2	2	12	6
38	9	2	10	4	6	10	12	10	14	4	12	8	6	10	8	6	2	10	2	14	6	6	4	6	6	4	4	12	6	16	4	14	12	8	8	8	8	10	0	8	12	2	4
39	17	10	6	8	14	2	4	6	6	12	4	8	14	2	8	6	6	10	10	6	10	6	8	14	10	12	12	4	2	8	12	6	8	4	12	16	4	2	8	0	4	10	4
40	13	14	10	8	18	6	8	2	10	8	8	4	18	6	12	6	10	14	10	2	14	10	12	10	6	12	16	4	6	12	16	10	12	4	16	20	8	2	12	4	0	14	8
41	11	4	8	6	4	12	10	12	12	6	10	10	4	8	6	8	4	8	4	16	4	4	2	8	8	6	2	14	8	14	2	12	10	10	6	6	6	12	2	10	14	0	6
42	13	6	6	4	10	6	8	6	10	8	8	4	10	6	4	2	2	6	6	10	6	2	4	10	6	8	8	8	2	12	8	10	8	4	8	12	4	6	4	4	8	6	0

Distance Matrix for Group Based Storage

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**
0	0	15	13	9	11	21	13	17	15	3	11	13	23	17	7	19	7	13	5	23	11	21	7	9	5	17	17	13	17	25	11	19	9	21	19	15	15	13	11	15	9	19	15
1	15	0	6	6	4	6	10	6	4	12	8	2	8	6	8	8	8	2	10	8	4	6	8	6	10	2	2	10	10	10	8	4	6	6	4	4	12	6	4	8	6	4	8
2	13	6	0	4	6	8	4	4	2	10	2	8	10	12	6	6	6	4	8	10	2	8	10	12	8	4	8	16	4	12	14	10	8	8	6	10	6	12	10	14	4	6	2
3	9	6	4	0	2	12	8	8	6	6	6	4	14	8	2	10	2	4	4	14	2	12	6	8	4	8	8	12	8	16	10	10	4	12	10	6	10	8	6	10	4	10	6
4	11	4	6	2	0	10	10	6	4	8	8	2	12	6	4	8	4	2	6	12	4	10	4	6	6	6	6	10	10	14	8	8	2	10	8	4	12	6	4	8	6	8	8
5	21	6	8	12	10	0	12	8	6	18	10	8	2	4	14	10	14	8	16	6	10	4	14	12	16	4	4	8	12	4	10	2	12	8	2	6	14	8	10	6	12	6	10
6	13	10	4	8	10	12	0	4	6	10	2	12	10	16	6	6	10	8	8	10	6	8	14	16	12	8	12	20	4	12	18	14	12	8	10	14	2	16	14	18	4	6	2
7	17	6	4	8	6	8	4	0	2	14	6	8	6	12	10	2	10	4	12	6	6	4	10	12	12	4	8	16	4	8	14	10	8	4	6	10	6	12	10	14	8	2	2
8	15	4	2	6	4	6	6	2	0	12	4	6	8	10	8	4	8	2	10	8	4	6	8	10	10	2	6	14	6	10	12	8	6	6	4	8	8	10	8	12	6	4	4
9	3	12	10	6	8	18	10	14	12	0	8	10	20	14	4	16	4	10	2	20	8	18	4	6	2	14	14	10	14	22	8	16	6	18	16	12	12	10	8	12	6	16	12
10	11	8	2	6	8	10	2	6	4	8	0	10	12	14	4	8	8	6	6	12	4	10	12	14	10	6	10	18	6	14	16	12	10	10	8	12	4	14	12	16	2	8	4
11	13	2	8	4	2	8	12	8	6	10	10	0	10	4	6	10	6	4	8	10	6	8	6	4	8	4	4	8	12	12	6	6	4	8	6	2	14	4	2	6	8	6	10
12	23	8	10	14	12	2	10	6	8	20	12	10	0	6	16	8	16	10	18	4	12	2	16	14	18	6	6	10	10	2	12	4	14	6	4	8	12	10	12	8	14	4	8
13	17	6	12	8	6	4	16	12	10	14	14	4	6	0	10	14	10	8	12	10	10	8	10	8	12	8	4	4	16	8	6	2	8	12	6	2	18	4	6	2	12	10	14
14	7	8	6	2	4	14	6	10	8	4	4	6	16	10	0	12	4	6	2	16	4	14	8	10	6	10	10	14	10	18	12	12	6	14	12	8	8	10	8	12	2	12	8
15	19	8	6	10	8	10	6	2	4	16	8	10	8	14	12	0	12	6	14	4	8	6	12	14	14	6	10	18	2	6	16	12	10	2	8	12	4	14	12	16	10	4	4
16	7	8	6	2	4	14	10	10	8	4	8	6	16	10	4	12	0	6	2	16	4	14	4	6	2	10	10	10	10	18	8	12	2	14	12	8	12	6	4	8	6	12	8
17	13	2	4	4	2	8	8	4	2	10	6	4	10	8	6	6	6	0	8	10	2	8	6	8	8	4	4	12	8	12	10	6	4	8	6	6	10	8	6	10	4	6	6
18	5	10	8	4	6	16	8	12	10	2	6	8	18	12	2	14	2	8	0	18	6	16	6	8	4	12	12	12	12	20	10	14	4	16	14	10	10	8	6	10	4	14	10
19	23	8	10	14	12	6	10	6	8	20	12	10	4	10	16	4	16	10	18	0	12	2	16	14	18	6	6	14	6	2	12	8	14	2	4	8	8	10	12	12	14	4	8
20	11	4	2	2	4	10	6	6	4	8	4	6	12	10	4	8	4	2	6	12	0	10	8	10	6	6	6	14	6	14	12	8	6	10	8	8	8	10	8	12	2	8	4
21	21	6	8	12	10	4	8	4	6	18	10	8	2	8	14	6	14	8	16	2	10	0	14	12	16	4	4	12	8	4	10	6	12	4	2	6	10	8	10	10	12	2	6
22	7	8	10	6	4	14	14	10	8	4	12	6	16	10	8	12	4	6	6	16	8	14	0	2	2	10	10	6	14	18	4	12	2	14	12	8	16	6	4	8	10	12	12
23	9	6	12	8	6	12	16	12	10	6	14	4	14	8	10	14	6	8	8	14	10	12	2	0	4	8	8	4	16	16	2	10	4	12	10	6	18	4	2	6	12	10	14
24	5	10	8	4	6	16	12	12	10	2	10	8	18	12	6	14	2	8	4	18	6	16	2	4	0	12	12	8	12	20	6	14	4	16	14	10	14	8	6	10	8	14	10
25	17	2	4	8	6	4	8	4	2	14	6	4	6	8	10	6	10	4	12	6	6	4	10	8	12	0	4	12	8	8	10	6	8	4	2	6	10	8	6	10	8	2	6
26	17	2	8	8	6	4	12	8	6	14	10	4	6	4	10	10	10	4	12	6	6	4	10	8	12	4	0	8	12	8	6	2	8	8	2	2	14	4	6	6	8	6	10
27	13	10	16	12	10	8	20	16	14	10	18	8	10	4	14	18	10	12	12	14	14	12	6	4	8	12	8	0	20	12	2	6	8	16	10	6	22	4	6	2	16	14	18
28	17	10	4	8	10	12	4	4	6	14	6	12	10	16	10	2	10	8	12	6	6	8	14	16	12	8	12	20	0	8	18	14	12	4	10	14	2	16	14	18	8	6	2
29	25	10	12	16	14	4	12	8	10	22	14	12	2	8	18	6	18	12	20	2	14	4	18	16	20	8	8	12	8	0	14	6	16	4	6	10	10	12	14	10	16	6	10
30	11	8	14	10	8	10	18	14	12	8	16	6	12	6	12	16	8	10	10	12	12	10	4	2	6	10	6	2	18	14	0	8	6	14	8	4	20	2	4	4	14	12	16
31	19	4	10	10	8	2	14	10	8	16	12	6	4	2	12	12	12	6	14	8	8	6	12	10	14	6	2	6	14	6	8	0	10	10	4	4	16	6	8	4	10	8	12
32	9	6	8	4	2	12	12	8	6	6	10	4	14	8	6	10	2	4	4	14	6	12	2	4	4	8	8	8	12	16	6	10	0	12	10	6	14	4	2	6	8	10	10
33	21	6	8	12	10	8	8	4	6	18	10	8	6	12	14	2	14	8	16	2	10	4	14	12	16	4	8	16	4	4	14	10	12	0	6	10	6	12	10	14	12	2	6
34	19	4	6	10	8	2	10	6	4	16	8	6	4	6	12	8	12	6	14	4	8	2	12	10	14	2	2	10	10	6	8	4	10	6	0	4	12	6	8	8	10	4	8
35	15	4	10	6	4	6	14	10	8	12	12	2	8	2	8	12	8	6	10	8	8	6	8	6	10	6	2	6	14	10	4	4	6	10	4	0	16	2	4	4	10	8	12
36	15	12	6	10	12	14	2	6	8	12	4	14	12	18	8	4	12	10	10	8	8	10	16	18	14	10	14	22	2	10	20	16	14	6	12	16	0	18	16	20	6	8	4
37	13	6	12	8	6	8	16	12	10	10	14	4	10	4	10	14	6	8	8	10	10	8	6	4	8	8	4	4	16	12	2	6	4	12	6	2	18	0	2	2	12	10	14
38	11	4	10	6	4	10	14	10	8	8	12	2	12	6	8	12	4	6	6	12	8	10	4	2	6	6	6	6	14	14	4	8	2	10	8	4	16	2	0	4	10	8	12
39	15	8	14	10	8	6	18	14	12	12	16	6	8	2	12	16	8	10	10	12	12	10	8	6	10	10	6	2	18	10	4	4	6	14	8	4	20	2	4	0	14	12	16
40	9	6	4	4	6	12	4	8	6	6	2	8	14	12	2	10	6	4	4	14	2	12	10	12	8	8	8	16	8	16	14	10	8	12	10	10	6	12	10	14	0	10	6
41	19	4	6	10	8	6	6	2	4	16	8	6	4	10	12	4	12	6	14	4	8	2	12	10	14	2	6	14	6	6	12	8	10	2	4	8	8	10	8	12	10	0	4
42	15	8	2	6	8	10	2	2	4	12	4	10	8	14	8	4	8	6	10	8	4	6	12	14	10	6	10	18	2	10	16	12	10	6	8	12	4	14	12	16	6	4	0

Distance Matrix for Group and COI Based Storage

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**
0	0	13	23	17	19	13	17	15	25	7	21	13	21	15	11	13	11	21	9	19	19	11	11	3	9	9	15	9	15	23	5	17	13	17	13	15	17	7	5	7	19	11	15
1	13	0	10	8	6	20	16	18	12	6	12	4	8	2	2	12	6	8	8	6	14	14	10	10	4	16	2	12	10	10	12	4	8	4	16	6	12	10	8	14	10	18	14
2	23	10	0	6	4	10	6	8	2	16	2	10	6	12	12	10	12	2	14	8	4	12	12	20	14	14	8	14	8	4	18	6	10	10	10	8	6	16	18	16	4	12	8
3	17	8	6	0	2	12	8	10	8	10	4	4	4	10	6	4	6	4	8	6	6	6	6	14	8	8	6	8	2	6	12	4	4	8	8	2	4	10	12	10	2	10	6
4	19	6	4	2	0	14	10	12	6	12	6	6	2	8	8	6	8	2	10	4	8	8	8	16	10	10	4	10	4	4	14	2	6	6	10	4	6	12	14	12	4	12	8
5	13	20	10	12	14	0	4	2	12	14	8	16	16	22	18	8	14	12	12	18	6	6	10	10	16	4	18	8	10	14	8	16	12	20	4	14	8	10	12	6	10	2	6
6	17	16	6	8	10	4	0	2	8	10	4	12	12	18	14	4	10	8	8	14	2	6	6	14	12	8	14	8	6	10	12	12	8	16	4	10	4	10	12	10	6	6	2
7	15	18	8	10	12	2	2	0	10	12	6	14	14	20	16	6	12	10	10	16	4	4	8	12	14	6	16	6	8	12	10	14	10	18	2	12	6	8	10	8	8	4	4
8	25	12	2	8	6	12	8	10	0	18	4	12	4	10	14	12	14	4	16	6	6	14	14	22	16	16	10	16	10	2	20	8	12	8	12	10	8	18	20	18	6	14	10
9	7	6	16	10	12	14	10	12	18	0	14	6	14	8	4	6	4	14	2	12	12	8	4	4	2	10	8	6	8	16	6	10	6	10	10	8	10	4	2	8	12	12	8
10	21	12	2	4	6	8	4	6	4	14	0	8	8	14	10	8	10	4	12	10	2	10	10	18	12	12	10	12	6	6	16	8	8	12	8	6	4	14	16	14	2	10	6
11	13	4	10	4	6	16	12	14	12	6	8	0	8	6	2	8	2	8	4	6	10	10	6	10	4	12	2	8	6	10	8	4	4	4	12	2	8	6	8	10	6	14	10
12	21	8	6	4	2	16	12	14	4	14	8	8	0	6	10	8	10	4	12	2	10	10	10	18	12	12	6	12	6	2	16	4	8	4	12	6	8	14	16	14	6	14	10
13	15	2	12	10	8	22	18	20	10	8	14	6	6	0	4	14	8	10	10	4	16	16	12	12	6	18	4	14	12	8	14	6	10	2	18	8	14	12	10	16	12	20	16
14	11	2	12	6	8	18	14	16	14	4	10	2	10	4	0	10	4	10	6	8	12	12	8	8	2	14	4	10	8	12	10	6	6	6	14	4	10	8	6	12	8	16	12
15	13	12	10	4	6	8	4	6	12	6	8	8	8	14	10	0	6	8	4	10	6	2	2	10	8	4	10	4	2	10	8	8	4	12	4	6	4	6	8	6	6	6	2
16	11	6	12	6	8	14	10	12	14	4	10	2	10	8	4	6	0	10	2	8	8	8	4	8	2	10	4	6	4	12	6	6	2	6	10	4	6	4	6	8	8	12	8
17	21	8	2	4	2	12	8	10	4	14	4	8	4	10	10	8	10	0	12	6	6	10	10	18	12	12	6	12	6	2	16	4	8	8	8	6	4	14	16	14	2	10	6
18	9	8	14	8	10	12	8	10	16	2	12	4	12	10	6	4	2	12	0	10	10	6	2	6	4	8	6	4	6	14	4	8	4	8	8	6	8	2	4	6	10	10	6
19	19	6	8	6	4	18	14	16	6	12	10	6	2	4	8	10	8	6	10	0	12	12	8	16	10	14	4	10	8	4	14	2	6	2	14	4	10	12	14	12	8	16	12
20	19	14	4	6	8	6	2	4	6	12	2	10	10	16	12	6	8	6	10	12	0	8	8	16	10	10	12	10	4	8	14	10	6	14	6	8	2	12	14	12	4	8	4
21	11	14	12	6	8	6	6	4	14	8	10	10	10	16	12	2	8	10	6	12	8	0	4	8	10	2	12	2	4	12	6	10	6	14	2	8	6	4	6	4	8	4	4
22	11	10	12	6	8	10	6	8	14	4	10	6	10	12	8	2	4	10	2	8	8	4	0	8	6	6	8	2	4	12	6	6	2	10	6	4	6	4	6	4	8	8	4
23	3	10	20	14	16	10	14	12	22	4	18	10	18	12	8	10	8	18	6	16	16	8	8	0	6	6	12	6	12	20	2	14	10	14	10	12	14	4	2	4	16	8	12
24	9	4	14	8	10	16	12	14	16	2	12	4	12	6	2	8	2	12	4	10	10	10	6	6	0	12	6	8	6	14	8	8	4	8	12	6	8	6	4	10	10	14	10
25	9	16	14	8	10	4	8	6	16	10	12	12	12	18	14	4	10	12	8	14	10	2	6	6	12	0	14	4	6	14	4	12	8	16	4	10	8	6	8	2	10	2	6
26	15	2	8	6	4	18	14	16	10	8	10	2	6	4	4	10	4	6	6	4	12	12	8	12	6	14	0	10	8	8	10	2	6	2	14	4	10	8	10	12	8	16	12
27	9	12	14	8	10	8	8	6	16	6	12	8	12	14	10	4	6	12	4	10	10	2	2	6	8	4	10	0	6	14	4	8	4	12	4	6	8	2	4	2	10	6	6
28	15	10	8	2	4	10	6	8	10	8	6	6	6	12	8	2	4	6	6	8	4	4	4	12	6	6	8	6	0	8	10	6	2	10	6	4	2	8	10	8	4	8	4
29	23	10	4	6	4	14	10	12	2	16	6	10	2	8	12	10	12	2	14	4	8	12	12	20	14	14	8	14	8	0	18	6	10	6	10	8	6	16	18	16	4	12	8
30	5	12	18	12	14	8	12	10	20	6	16	8	16	14	10	8	6	16	4	14	14	6	6	2	8	4	10	4	10	18	0	12	8	12	8	10	12	2	4	2	14	6	10
31	17	4	6	4	2	16	12	14	8	10	8	4	4	6	6	8	6	4	8	2	10	10	6	14	8	12	2	8	6	6	12	0	4	4	12	2	8	10	12	10	6	14	10
32	13	8	10	4	6	12	8	10	12	6	8	4	8	10	6	4	2	8	4	6	6	6	2	10	4	8	6	4	2	10	8	4	0	8	8	2	4	6	8	6	6	10	6
33	17	4	10	8	6	20	16	18	8	10	12	4	4	2	6	12	6	8	8	2	14	14	10	14	8	16	2	12	10	6	12	4	8	0	16	6	12	10	12	14	10	18	14
34	13	16	10	8	10	4	4	2	12	10	8	12	12	18	14	4	10	8	8	14	6	2	6	10	12	4	14	4	6	10	8	12	8	16	0	10	4	6	8	6	6	2	2
35	15	6	8	2	4	14	10	12	10	8	6	2	6	8	4	6	4	6	6	4	8	8	4	12	6	10	4	6	4	8	10	2	2	6	10	0	6	8	10	8	4	12	8
36	17	12	6	4	6	8	4	6	8	10	4	8	8	14	10	4	6	4	8	10	2	6	6	14	8	8	10	8	2	6	12	8	4	12	4	6	0	10	12	10	2	6	2
37	7	10	16	10	12	10	10	8	18	4	14	6	14	12	8	6	4	14	2	12	12	4	4	4	6	6	8	2	8	16	2	10	6	10	6	8	10	0	2	4	12	8	8
38	5	8	18	12	14	12	12	10	20	2	16	8	16	10	6	8	6	16	4	14	14	6	6	2	4	8	10	4	10	18	4	12	8	12	8	10	12	2	0	6	14	10	10
39	7	14	16	10	12	6	10	8	18	8	14	10	14	16	12	6	8	14	6	12	12	4	4	4	10	2	12	2	8	16	2	10	6	14	6	8	10	4	6	0	12	4	8
40	19	10	4	2	4	10	6	8	6	12	2	6	6	12	8	6	8	2	10	8	4	8	8	16	10	10	8	10	4	4	14	6	6	10	6	4	2	12	14	12	0	8	4
41	11	18	12	10	12	2	6	4	14	12	10	14	14	20	16	6	12	10	10	16	8	4	8	8	14	2	16	6	8	12	6	14	10	18	2	12	6	8	10	4	8	0	4
42	15	14	8	6	8	6	2	4	10	8	6	10	10	16	12	2	8	6	6	12	4	4	4	12	10	6	12	6	4	8	10	10	6	14	2	8	2	8	10	8	4	4	0