



Graduate Theses, Dissertations, and Problem Reports

2018

Flexible high-density puzzle storage systems

Ehsan Shirazi

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Recommended Citation

Shirazi, Ehsan, "Flexible high-density puzzle storage systems" (2018). *Graduate Theses, Dissertations, and Problem Reports*. 3992.

<https://researchrepository.wvu.edu/etd/3992>

This Problem/Project Report is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Problem/Project Report in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Problem/Project Report has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Flexible High-Density Puzzle Storage Systems

By

Ehsan Shirazi

Problem Report submitted to the Benjamin M. Statler College of Engineering and Mineral

Resources at West Virginia University In partial fulfillment of the degree of

Master of Science

in

Industrial Engineering

Committee Members:

Kenneth Currie, Ph.D., Chair

Leily Farrokhvar, Ph.D.

Thorsten Wuest, Ph.D.

Department of Industrial and Management Systems Engineering

Morgantown, West Virginia

June 2018

Abstract

In a traditional storage system, when one of the mechanical or electrical parts or the software fails to operate correctly, it may lead to the failure of the entire system imposing huge costs to the system (Gue et al., 2014). However, in a grid storage network, if a grid fails to operate, the system continues to operate. Therefore, failure of a system component will not jeopardize the whole system functionality. As such, reliability is one of the key advantages of grid storage networks.

Another benefit of grid storage systems is flexibility. Various systems such as GridFlow, GridPick, GridStore, are developed that consist of multiple autonomous conveyors to store and move items in different directions. A unit module is capable to work independently. To retrieve an item, empty modules open a temporary aisle to move the item to the retrieval point.

This study developed a tote retrieval algorithm for high-density storage systems (HDSS). The algorithm is capable of retrieving totes from all sides of a storage puzzle. This is novel since methodologies of prior studies were confined to retrieving items from a single grid or a single side of a puzzle. The algorithm was decentralized and agent-based where each grid acts as an agent. The study undertook developing algorithms that could minimize the number of movements in the network for tote retrievals and to prevent deadlocks. Deadlock prevention algorithms are capable to resolve a diverse range of situations that cause network deadlocks.

An object-oriented software program was developed that implemented the algorithms. The software tool simulated puzzles of various sizes with different number of escorts available for retrieving requested totes. Thousands of iterations of the puzzle configurations were resolved for the analysis. It was found that incremental increase in the number of escorts in the puzzle reduces the number of movements for retrieving totes. However, depending on the puzzle size, there were increments that additional escorts had minimal impact. It was also found that average retrieval movements for 3 totes increases logarithmically with the number of cells in a puzzle.

To validate the methodology and its software implementation, results from the simulation were compared with the results from other studies with mathematical solutions. The program replicated the optimum number of movements for a puzzle that contained one escort and retrieved one tote.

Keywords: Puzzle-Based Storage, Puzzle Network, Decentralized Network, Agent-based Modeling, Storage System, 15-Puzzle

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION	1
1.1 Background.....	1
1.2 Motivation.....	2
1.3 Traditional Inventory	5
1.4 Puzzle Networks	5
1.5 Structure of Report	7
CHAPTER 2. LITERATURE REVIEW	8
2.1 Introduction to Puzzle Based High Density Storage System.....	8
2.2 Related Literature	9
2.3 Summary and Literature Gap.....	15
CHAPTER 3. Definitions and Methodology.....	18
3.1 Network Definitions	18
3.2 Retrieval Scenarios	19
3.2.1. Assigning white cells to specific areas of the puzzle	19
3.2.2. Requiring at least one empty cell in each column and row	20
3.2.3. Assigning a white cell to a requested item through entire retrieval	20
3.2.4. White cells assigned based on nearest distance after every movement.....	21
3.3. Methodology Definitions and Assumptions	21
3.4. Methodology Flowcharts	30

3.5. Puzzle Network Retrieval Examples	32
CHAPTER 4. Analysis and Results.....	38
4.1 Simulation Results	38
4.2 Comparing Simulation Results with Previous Studies	48
CHAPTER 5. Conclusion and Future Work.....	53
5.1 Accomplishments of this research and contributions	53
5.2 Conclusion	53
5.2 Recommendations for Future Work	55
References.....	56

List of Figures

FIGURE 1: DESIGNED PUZZLE WITH ASSIGNED WHITE CELLS TO THE SPECIFIC SUBAREAS OF THE BOARD	20
FIGURE 2: ONE MOVEMENT IN THE BOARD	21
FIGURE 3 OVERALL STEPS IN SOLVING A PUZZLE.	23
FIGURE 4: OPTIMAL SIDE OF REQUESTED ITEM IDENTIFICATION	24
FIGURE 5: SEARCHING STEPS FOR SEIZING AN EMPTY CELL	25
FIGURE 6: SCHEMATIC OF 3-MOVES SCENARIO.....	26
FIGURE 7: SCHEMATIC OF 5-MOVES SCENARIO.....	27
<i>FIGURE 8: FIRST DEADLOCK EXAMPLE: COMPETING BLACK CELLS TO SEIZE THE SAME WHITE CELL AT EQUAL DISTANCE.....</i>	<i>29</i>
<i>FIGURE 9: SECOND DEADLOCK EXAMPLE: BLACK CELL SEIZING 2 WHITE CELLS AT EQUAL DISTANCE FROM THE OPTIMAL SIDE.</i>	<i>29</i>
FIGURE 10: THIRD DEADLOCK EXAMPLE: COLLIDING BLACK CELLS BEING RETRIEVED FROM OPPOSING SIDES WHILE SEIZING DIFFERENT WHITE CELLS.	29
FIGURE 11- OVERALL PUZZLE NETWORK ALGORITHM	31
FIGURE 12- OVERVIEW OF DEADLOCK PREVENTION ALGORITHM	32
FIGURE 13: ILLUSTRATION OF HOW PUZZLE WORKS.....	34
FIGURE 14: ILLUSTRATION OF HOW PUZZLE WORKS	35
FIGURE 15: CHOOSING AN ESCORT AND RETRIEVAL PATH	36
FIGURE 16: ILLUSTRATION OF OPTIONS FOR RETRIEVING A REQUESTED ITEM.....	37
FIGURE 17: STEPS IN CALCULATING SIMULATION RESULTS.	39
FIGURE 18: AVERAGE NUMBER OF MOVEMENTS FOR RETRIEVED ITEMS IN EVERY ITERATION FOR 6*6 PUZZLE WITH 18 WHITE CELLS.	40
FIGURE 19- SIMULATION RESULTS FOR 6×6 PUZZLE	41

FIGURE 20: AVERAGE NUMBER OF MOVEMENTS FOR RETRIEVED ITEMS IN EVERY ITERATION FOR 12×12 PUZZLE WITH 18 WHITE CELLS.	43
FIGURE 21: SIMULATION RESULTS FOR 12×12 PUZZLE.....	43
FIGURE 22: AVERAGE NUMBER OF MOVEMENTS FOR RETRIEVED ITEMS IN EVERY ITERATION FOR 20×20 PUZZLE WITH 18 WHITE CELLS.	44
FIGURE 23: SIMULATION RESULTS FOR 20×20 PUZZLE.....	44
FIGURE 24: AVERAGE NUMBER OF MOVEMENTS FOR RETRIEVED ITEMS IN EVERY ITERATION FOR 25×25 PUZZLE WITH 18 WHITE CELLS.	46
FIGURE 25: SIMULATION RESULTS FOR 25×25 PUZZLE.....	46
FIGURE 26: IMPACT OF PUZZLE SIZE ON AVERAGE RETRIEVAL MOVEMENTS (ARM) WITH ONE WHITE CELL.....	47
FIGURE 27: IMPACT OF NUMBER OF PUZZLE CELLS ON AVERAGE RETRIEVAL MOVEMENTS (ARM) WITH ONE WHITE CELL.....	48
FIGURE 28- ILLUSTRATIVE EXAMPLE OF GUE AND KIM (2007) BLACK CELL LOCATION REFERENCING METHOD.....	49
FIGURE 29: COMPARISON IN AVERAGE NUMBER OF MOVEMENTS BETWEEN RANDOM AND INPUT/OUTPUT DISTRIBUTIONS	51
FIGURE 30: COMPARISON OF I/O DISTRIBUTION	52
FIGURE 31: COMPARISON OF RANDOM DISTRIBUTION.....	52
FIGURE 32: SEVERAL LEVELS PUZZLE SYSTEM.....	55

List of Tables

TABLE 1: ASSOCIATED NUMBER OF MOVEMENTS BASED ON SELECTED ESCORT AND PATH FOR EXAMPLE 2.	37
TABLE 2: COMPARISON OF SOFTWARE PROGRAM SIMULATION RESULTS WITH MATHEMATICAL OPTIMUM SOLUTION OBTAINED BY GUE AND KIM (2007).....	49
TABLE 3: COMPARISON BETWEEN DIFFERENT CONFIGURATIONS OF EMPTY SPOTS IN TAYLOR ET AL. (2008) AND CURRENT RESEARCH	51
TABLE 4: WHITE CELLS INDIFFERENCE INTERVALS FOR SIMULATED PUZZLE SETTINGS	54

Acknowledgement

I am thankful to all of those individuals with whom I have had the pleasure to grow up with in these years.

I would like to thank my wife, Atefeh Malekinezhad, who supported me with all means.

I would like to thank my advisor Dr. Currie for his patience and for guidance he provided me with this research.

I would like to thank the members of my committee; Dr. Leily Farrokhvar and Dr. Thorsten Wuest. They have provided me great guidance and taught me a great deal about scientific research.

CHAPTER 1. INTRODUCTION

1.1 Background

According to Krühn et al (2016) when an intralogistics system is installed, the overall system is projected to be used for many years. An intralogistics system is comprised of different subsystems that are integrated to each other. Modifying or upgrading a part of a subsystem usually causes a major part, or the entire system, to stop working. Recent developments in commercial sales have focused on lowering product life-cycle and mass-customization. These developments affected the manufacturing or warehousing practices and led them toward use of autonomous and decentralized technologies. These advances in manufacturing and warehousing resulted in more flexibility in the entire system.

Computerized systems are considered agent-based systems because they have the ability to change or be updated very fast and work independently. Another main feature of agent-based models is a message-passing attribute. Computers, controls and sensors are being used to incorporate intelligence into such systems. (Jennings et al, 2001).

A high-density storage systems (HDSS) refers to a situation where items in a warehouse are stored deeply on the shelves or on top of each other. Gue et. al. (2006 and 2014) presented and elaborated on the concept of decentralized puzzle based HDSS. In the preliminary concept, they introduced a puzzle network that included a single input and a single output location (cells) that could be used for replenishing and retrieving items. The puzzle network could also include a single or multiple empty spots. Empty spots are used to retrieve an object from the puzzle. Puzzle network was inspired from the 15-puzzle game commonly played by children. Taylor et al. (2008) studied the effect of the empty spot in a single input/output spot in a 4×4 puzzle with different system settings. In Gue et al.'s model (2014), items are replenished from the top of the

puzzle, are stored in the middle and are retrieved from the bottom of the puzzle. Deadlocks in puzzle networks are defined as situations where the system algorithm is not capable to advance the next move of the puzzle. Many researchers have endeavored to develop algorithms that prevent deadlocks. In Gue's model (2014), an entering item from the top negotiates with the row below to seize an empty cell. This continues until the item reaches its planned storage location known as home row. Requested items in the puzzle use the same logic to be retrieved from the bottom of the board.

1.2 Motivation

Material Handling (MH) is one of the major parts of the economy that should be taken into account. Developed countries such as the United States are working to lower the cost of MH. This has a major effect on remaining competitive in today's global marketplace. One of the important components of material handling systems is retail industry. Retail companies, such as Amazon, are in severe challenge for changing and updating their MH systems. Some of these retail companies are seeking models and techniques that can deliver goods or services in the same day or even faster. They not only seek to satisfy demand as quick as possible, but also at a lower cost. Implementing emerging technologies is critical to their competitive advantage. One of the major motivations of this research is to analyze and suggest ways that these companies can use to meet demands faster and at lower cost. This study examines usage of modern autonomous agent-based conveyor system to support high density, high throughput storage/retrieval. The study developed new algorithms and improved some of the existing ones for these systems, and implemented these algorithms into an object oriented software program.

The Material Handling Institute (MHI, 2014) released its first version of a material handling technology roadmap in 2014. In this document, ten major technology trends in material handling

are recognized and explained. Robotics and automation, sensors and internet of things (IoT) are among identified technology trends in material handling systems closely related to autonomous agent-based conveyor systems. This signifies the importance and timeliness of this research.

Sensor technology enables companies to incorporate intelligence into the systems and machinery they use. Conveyors may conduct specific tasks autonomously, which represents a decentralized system, as opposed to a system where all tasks are dictated to conveyors from a supervisory level, which represents a centralized system. A decentralized system runs on sophisticated algorithms and data. The data is provided and collected concurrently in the system, or from the past decisions without any human interaction. The idea of such a system can be expanded by the concept of the Internet of Things (IoT) where human interference is further reduced. The challenge of the MH industry to implement the IoT concept, similar to many other industries, is to tie these capabilities together to achieve efficient, high-throughput material movement and storage.

The other projected trends presented in the MHI technology roadmap are also relevant to high-density storage/retrieval systems. For instance, urbanization is identified as a trend affecting future MH systems. Urbanization requires goods and services to be provided in densely populated areas. Among the challenges imposed by urbanization, cost of storage facility land is a key factor. High-density storage systems is a viable solution to this challenge that strives to maximize the use of available space within the storage facility.

Decisions regarding implementation of a particular MH system is weighed against advantages and disadvantages among classical parameters such as flexibility, operational and investment costs, quality and reliability of each system as well as the time required to replenish or retrieve items from the system. Generally, an automated system requires high initial

investment costs whereas a manual system necessitates high operational costs. Automated systems have high quality and reliability but are less flexible. Manual systems are more flexible in handling and transport but more prone to quality and reliability issues. The ever-increasing pace of business requires new technological solutions. To be successful, companies should enhance their material handling technologies to the ones that are more customer based and highly flexibility that meets high quality of services (Schmidt et al., 2009).

As noted previously, this research focuses on a particular type of automated systems known as puzzle based high-density storage network. The goal in these storage systems is to maintain high flexibility while simultaneously trying to minimize the number of movements required to replenish or retrieve items in the system. When comparing older conveyer systems with these puzzle based storage systems, the following shortcoming were identified by Furmans (2010):

- Change to a small part of the traditional fixed conveyor system could be very costly. Since all of the mechanical, electrical and software parts of the system are integrated, if some parts are updated, it will most likely require that all other components be updated to the new system requirements.
- If a small part of the system fails, the entire system may be required to shut down for trouble shooting which can impose a tremendous amount of cost. It also imposes a lot of idle time to the whole warehouse.
- Traditional conveyor systems are centralized networks and require a large capital investment. However, high-density puzzle based systems are decentralized and agent based, and are customizable and adaptable.
- Fixed conveyor systems are not easily adaptable. When the storage faces fluctuations in demands, an agent-based network, can easily adapt to the new demand.

In summary, a well-designed storage system must store unit loads in the most efficient way to save space while allowing reduced retrieval and replenishment time. Automated systems are flexibility and can meet high performance criteria. In decentralized automated systems, a message passing protocol is implemented between grids.

1.3 Traditional Inventory

In traditional storage systems, items are stored in a block layout where all of the same units are stacked upon one another, or are stored in shelves with several unit loads deep. This limits the storage space to half-utilized racks. In addition, when multiple Stock Keeping Units (SKUs) are stored in the same block or shelf, it becomes difficult to retrieve a requested item that is lodged in the middle. These systems are usually based on a “first in first out” inventory policy. Therefore, retrieving a requested item requires that workers and machines retrieve unnecessary items or replenish them.

The goal of traditional warehouse systems is to maintain inventory at the economic level to satisfy two objectives. The first objective is to satisfy demands in a timely manner. If the company is not able to reach this goal, it loses its customer. The second objective is to keep the inventory level at an economic level. If the company accumulates goods in the warehouse beyond its customers’ demand, it ties up precious capital that could be optimized to generate profit.

1.4 Puzzle Networks

Gue et al. (2006) studied puzzle based HDSS that can store high number of items in a rectangular space. This space is similar to a chessboard in which each item can be stored in each cell. For replenishing or retrieving an item, the puzzle network needs empty spots to direct the

requested item to the exit point. There are many situations that using a high-density storage system is beneficial as illustrated in the following examples:

1) When stored goods are in high demand and storage is only temporary. A good example is a cross-docking warehouse where orders are constantly being assembled as goods arrive, and then are stored in a storage grid.

2) When the available space is very limited, such as in cargo ships. In addition, when warehouse space is very costly or sparse such as in dense metropolitan areas. HDSS may be the only viable option for storage in these situations.

3) In addition, using HDSS allows building different storage layouts that best conform to requirements. For example, a square or rectangular network can be set and different assembly or manufacturing machines can be put in different sides of it. In order to assemble different objects stored in the decentralized network, items make movements based on the defined sequence to get to the proper machine station, and then are replenished. New items are released for the next operation or are replenished from the network when they are completed.

4) Also, these types of networks are ideal for seasonal warehouse fluctuations. For example, since independent agents for building up a decentralized network are being used, different layouts of the network with different sizes can be built. For instance, when market fluctuations create the need for additional storage, traditional warehouse solutions tend to keep a larger inventory according to the highest seasonal demand Seibold (2013).

5) Many warehousing and distribution companies receive complaints from customers that the goods they received were damaged. One reason for damage is that in traditional warehousing,

goods and totes are stored on top of each other to optimize storage density. Using HDSS reduces the likelihood of such damage to totes while in storage.

1.5 Structure of Report

Chapter 2 presents a literature review relevant to the study. Special attention is paid to topics such as agent-based simulation, problem difficulty, automation systems in warehouse and storage area, automated storage and retrieval systems, puzzle storage systems, decentralized algorithms and so on.

Chapter 3 presents the research methodology that has been used in this study. It identifies the gaps in prior research and presents the study's effort to address them.

Chapter 4 presents different scenarios used for message passing protocols and various settings of the puzzle network. Also, the chapter identifies the details of the functions used in the model and system behavior explained with respect to simulation with details.

Chapter 5 presents conclusions and includes some recommendations for future work.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction to Puzzle Based High Density Storage System

The puzzle based high-density storage system (HDSS) can be used in areas such as manufacturing, inventory or a combination of them. In an inventory environment, for example, the goal for using this grid network is to eliminate unnecessary movements when an item has been requested. The solution metric or the performance criteria are minimization of time or the number of movements by the system for retrieving or replenishing an item. When an item is requested, interfering stored goods should sidestep to make an empty spot for the requested item. In previous studies, items only replenish from the top row and store on a home row, such as in GridStore (Gue, 2014). In addition, items can only be retrieved from the bottom of the puzzle network. Interfering items move to the left and right to step away from the path of the entering or leaving items.

Other varieties of puzzle networks are introduced in the literature as well. Examples include a puzzle that can replenish and retrieve from one spot with only one empty (white) cell, or a puzzle that can replenish and retrieve from only one spot with several empty (white) cells. Another example includes a puzzle storage with racks while using Automated Guided Vehicles (AGVs) for retrieving and replenishing items.

The puzzle introduced in this study brings more flexibility into the network. It can replenish and retrieve from all sides. When an item is requested from the puzzle, it employs an empty cell as the first phase. In the second phase, puzzle makes a movement. This continues until the requested item exits from the puzzle at the desired location.

2.2 Related Literature

Storage systems/networks have a vital role in the manufacturing and warehousing industry. The cost associated with storing items and the amount of time required to retrieve and replenish them are important factors in managing effective and efficient storage systems/networks. One way to decrease the cost is to utilize the space as efficient as possible, which leads to the concept of high-density storage systems (HDSS). HDSS is usually achieved by increasing the area dedicated to storage through reducing or eliminating the dimensions of the aisles used to retrieve items.

In effective management of HDSS, human interference must be minimized. This resulted in the development of storage systems that are automated in certain functions. According to Berg and Gademann (2000) automated storage and retrieval systems (ASRSs) are currently being utilized extensively in the industry. ASRS is a technology that supports warehouse or manufacturing companies in storing and retrieving products automatically without human interference. ASRS includes a component that travels through the storage aisles to both retrieve and replenish items. The items are stored in racks and shelves that are built in every aisle of the HDSS. Each shelf can accommodate one or many items. Therefore, one or more units of an item can be stored in a spot on the shelf. This creates more space for usage. Berg and Gademann (2000) studied the impact of different policies that could be implemented for control and storage of the items. Policies researched included various sequencing of the items storage and retrieval requests. Also, the impact of low space utilization on item retrieval time was studied.

According to Furmans et al. (2010), one main weakness of traditional automatic handling systems is that they are not simple. When shelves or racks are installed and conveyors are fixed in place, making even an insignificant change is difficult or impractical. Recent progress in

modern-day material handling systems has resulted in more capabilities and lower initial cost for basic system modules such as controllers and sensors. The authors presented new designs for decentralized systems and developed prototypes for so-called "Plug and Work" systems. These systems are simple to reconfigure. SmartRack system, FlexConveyor, KARIS and Puzzle-based storage systems are examples of reconfigurable systems described by the authors. Gebhardt Company has led the developments of FlexConveyor, and Karlsruhe Institute of Technology (KIT) developed KARIS systems

In another research, Rohit et al. (2010) studied the performance of retrieval time associated with puzzle-based storage systems. Previous works were focused on the analysis of the retrieval time when a puzzle has only one entrance and exit points with one or more escorts (empty cells). While in previous studies, empty cells were placed near the entry point, Rohit et al studied the performance of the system when the empty cells were distributed randomly in the puzzle network. The system performance was measured and analyzed in terms of retrieval-time. They first measured the system performance when only one escort existed and was placed randomly in the puzzle network. They used integer programming technique for the analysis. Also, in another analysis, two escorts were used in the puzzle network; one was placed at the network entry point and the other was placed randomly inside the puzzle. Authors claim that results are valid for puzzles of any size. Also, an integer programming formulation has been presented at the end of this research.

Seibold (2011) defines GridFlow as a modular storage system that brings flexibility into intralogistics system. This technology, GridFlow, is designed and developed at Auburn University. In GridFlow, several FlexConveyors are connected to each other in a network. FlexConveyor is an autonomous conveyor module that is equipped with a controller and decides

whether to pass or retain a load that is stored on top of it. A set of these independent conveyors build a multi-agent network model with every conveyor serving as an agent. To reduce the transactions between agents, "Message-Passing-Protocol" is established. Using agent based network model in combination with message passing protocol reduces the complexity of the system and brings more flexibility into the system. The protocol dictates whether a particular agent keeps or passes the load. Authors implemented different scenarios with different processing volumes to measure the performance of the designed system. Also, it was shown that the network was deadlock free even if one or more modules fail. When one or more modules fail, the system continues with reasonable performance without any interruption. This allows regular maintenance to fix the failures rather than requiring an emergency maintenance.

Furmans et al. (2010) developed policies for a decentralized modular conveyor that prevents deadlock in the network and numerates the conditions that could cause deadlocks. They indicate that deadlocks could be avoided if the system design assure that at least of those conditions are not met.

Alfieri et al. (2012) proposed a different puzzle model which he claims can be implemented through a cheaper technology than its predecessors. In the proposed model, he used motorized shelves. A set of Automated Guided Vehicles (AGVs) are transmitted to the requested item location to support the movements. In their model, AGVs are the agents and replace FlexConveyors. Models based on AGVs use more complex scenarios than FlexConveyors to retrieve and replenish items. In a puzzle-based network, message-passing protocol is employed by AGVs to make decisions while moving loads. However, in such a system AGVs should be set in a concordant way to retrieve or replenish items. In this work, a new heuristic algorithm is proposed, and validated with a simulation. While the study dealt with a fixed list of picking

demands to be satisfied in a minimum period, it is recognized that handling a dynamically changing list of retrieval demands is more complex.

Gue et al. (2013) developed GridStore, which is a decentralized and deadlock-free puzzle network. Loads are replenished and retrieved from the top and bottom of the puzzle. Each FlexConveyor is modeled as an agent. It communicates using a message passing protocol with its neighbors. Replenishing items location is determined before entering into the storage system, which means loads transfer to the home row. Finally, if an item is requested, it would be retrieved from the bottom of the puzzle.

Decentralized material handling is developed to boost flexibility in the network. Seibold et al. (2013) studied such a system using FlexConveyor as a significant player in the decentralized-control system. FlexConveyor is an autonomous module that utilizes a set of modules to establish a decentralized network. This study shows that FlexConveyor is suitable to be used for different network layout. High-density storage networks need high output while limited space is provided for sorting goods. Different analysis and modeling techniques presented in the national transportation network were used. These techniques are similar to network/system analysis. The analysis was used to determine bottlenecks and specify results based on an optimization algorithm. Different sorting layouts were examined, such as trapezoid, square, circular, and ring. Various sorting factors were included in the model to measure network performance based on elements such as space, number of modules, shortest path, the risk of deadlock, etc. Findings show that a square set up needs the lowest amount of space, has the shortest path length and needs 1.5 times less direction changes than other settings; however, the chance of deadlock is higher. Findings of this study were compared with the results of a discrete simulation model. It was concluded that a high-density storage network is suitable for sorting good.

One highly related issue to high-density storage systems is buffering issue. Sohrt et al. (2014) studied the impact of storage buffers on system flexibility and capacity. Storage buffering is necessary as it links the gap between transportation and storage. Two conceptually different buffers were designed – static and dynamic buffers. Both buffer algorithms were developed to evade deadlock and collisions. It was concluded that a dynamic buffer increases the flexibility and capacity of the network at the price of higher energy consumption.

Uludag (Dissertation, 2014) designed a high-density decentralized puzzle based order picking network, named GridPick. GridPick consists of modular conveyors. It uses message passing protocol, negotiation algorithm in a decentralized system. GridPick was originally designed to retrieve totes from one side of the puzzle network. Then the system was modified to retrieve items from two sides. PetriNets software was used to simulate the storage system and to identify situations that may lead to deadlocks. The study results presented retrieval time for the original system that retrieves totes from one side as well as the retrieval time for the modified system that retrieves totes from two sides. The study includes simulation results for various network sizes.

Krühn et al. (2016) suggested a small scale and multi-directional modules that create a conveyor matrix. When a large number of modules are put together, it acts as a decentralized network that can carry out retrieval and replenishment movements. Each module has its controller and sensors that allow the module to communicate with its neighbors through a message passing protocol. The mechanical part of module components and its potential deadlock-free algorithms are presented in this research. Algorithms focus on local decisions, which lead to prevent deadlock locally and globally. An experiment was conducted on a prototype of a puzzle to examine conveyance performance of the system. It was concluded that

building a real-life scale of such a system is possible with these small-scale modules. Also, it was found that employing such a network reduces complex conveyance in large systems to local and solvable problems.

Gue et al. (2012) suggested using a decentralized grid-based puzzle network to feed a palletizing robot. The approach is built upon a flexible and scalable autonomous control algorithm in which each grid communicates with its neighbors to do sequencing. In the end, the study identified that a puzzle with fewer columns has better performance. To improve performance, the authors suggested that the ratio of the rows to the columns should at least be ten.

Based on Derhami et al. (2016) one of the widely used and expensive methods for storing items in manufacturing and warehousing is Block Stacking. This study investigated optimum warehouse space using a simulation model. The study identified an optimal lane-depth to maximize the space utilization for various production rates. To validate the study results, experimental scenarios were developed and tested. The analysis results indicated that the models were robust and accurate.

While prior studies focused on retrieval of one single load, Mirzaei et al. (2017) studied various arrangements of loads retrieval to minimize system retrieval time. The study first focused on retrieving two loads from each grid. Then, it expanded to retrieving multiple loads. The results showed that a double-loaded puzzle saved seventeen percent in retrieval time. The authors suggested using multiple-loads to decrease retrieval times.

2.3 Summary and Literature Gap

The idea of a puzzle network came from two kids' single player games, Rush Hour, and the 15-puzzle game. At first glance, these two games sound to be very easy to solve, but they are categorized as difficult problems. Hearn and Demaine (2005) showed that multiple motion planning problems are PSPACE-Hard. Also, they concluded that all moving blocks puzzles are PSPACE-Hard. Flakeet. al. (2002) showed that "Rush Hour" is PSPAE-Complete. The aim of Rush Hour is to slide the red car out of the board by moving other cars while the number of movements in the board is minimum. The constraint for playing this game is that vehicles are limited in the lanes. Vehicles can move left, right, up and down based on the lanes. Another classic game, 15-Puzzle, is a game with 15 numbered blocks from one to 15. The goal of the game is to put numbers in order with minimum number of movements.

Gue et al. (2007) were inspired from the 15-Puzzle game, and built high-density storage in which each cell, representing an agent, is a machine that can store totes on top of it. This design eliminates the need for dedicated aisles. The system was named GridFlow. Authors conducted experimental analysis with various settings including one or multiple empty cells. In all scenarios, only one cell was assigned as entry or exit point.

Gue (2010) introduced several material handling equipment that can be used in a HDSS. FlexConveyor is a unit-size modular conveyor that can store an item on top of it. It can send an item in four directions. This device is able to be connected with its adjacent modules to send or receive messages. These messages lead FlexConveyor to decide whether to retain its item or send it in one of four directions. FlexConveyor provides scalability and adaptability required for building the HDSS in practice.

Another device developed for implementation in HDSS is Karis. Karis is an extension of FlexConveyor with more advanced features. Karis combines the technology embedded in AGV systems and the conveyor system.

In Gue's team research in 2007 through 2010, the main objective was analytical determination of retrieval time when items can be retrieved or replenished from one spot with one or several escorts. The escorts were all at input/output (I/O) point. Rohit et al. (2010) further expanded Gue's research by randomly placing escorts in the puzzle and measured the retrieval time of an item.

Gue and Furmans (2011) introduced decentralized rectangular high-density storage in which items could be replenished and retrieved from top and bottom of the puzzle, respectively.

In another study, Gue et al. (2012) showed an application of decentralized puzzle based storage in cartoon sequencing. The study shows how the change in the number of escorts changes the performance of the system. Moreover, authors suggested the aspect ratio, defined as the number of rows divided by the number of columns, to be at least 10 for better system performance. Furmans et al. (2013) illustrated that a puzzle-based network may avoid deadlock in the event of one or more grids failure.

Gue et al. (2014) also developed GridPick as a continuation of his previous studies. In this study, he proved that the system is deadlock free and the results for different configurations of the system have been discussed. Finally, Hao and Gue (2016) describe another example of GridPick where "A decentralized control for a two-sided grid-based puzzle Rail-Rail Physical Internet hub" is explained. The puzzle introduced in this paper brings more flexibility into the

network puzzle. They also investigated situations where the system may face deadlocks and devised algorithm to avoid them.

In previous studies, items could enter or exit from one I/O point. In more recent studies, items could enter from one side of the puzzle and exit from another side. In this research, items can replenish and retrieve from all sides and from all points. When an item is requested from the puzzle, it employs an empty cell, and then it makes a movement according to the algorithms developed. This cycle continues until the requested item exits from the puzzle at the desired location.

Overall, compared to previous studies, findings from this research offer a more flexible puzzle network since loads stored in the puzzle can be retrieved from all sides. The previous network replenishing and retrieving from two sides of the puzzle, every row must keep an empty cell to prevent deadlock, however, the network presented in this study requires only one empty cell to function. This allows increasing the storage density of the network.

CHAPTER 3. DEFINITIONS AND METHODOLOGY

3.1 Network Definitions

- Board or Puzzle

A board or puzzle consists of agents represented by every cell. This is a concept inspired by the 15-puzzle game and Rush Hour Gue et al. (2014). Each cell is an agent.

- Agent

Agents are cells that establish the board. An agent is a plug and play autonomous machine that can store an item, or it can be empty. An agent is responsible to decide whether to keep or pass the tote stored on top of it. The tote on a cell can be passed forward, backward, left or right based on defined logic of the agent and communication between agents.

- Empty Cell/Module/Grid (Escort)

If no item is stored in a cell, it is considered an empty module/grid, also referred to as a white cell or escort. The number of empty spaces in a puzzle network can differ based on the performance expected from the board. For example, Gue et al. (2014) suggested maintaining a fixed number of white cells in every row of the board to avoid deadlock. Because the white cells are needed for movement of unit loads, they are also referred to as escorts. Escorts continuously move along with the requested loads for retrieval. The number of escorts in a puzzle and the policy of their assignment to requested loads affect the efficiency of the system.

- Stored Item

Stored items are goods, totes (items), pallets, or unit loads that can be stored in a cell. In this research, stored items are presented in gray color on board diagrams.

- Requested Item

Stored items can be called and requested for retrieval. In this research, requested items are shown as black in board diagrams. White or escort cells accompany black cells while being retrieved from the puzzle.

- Deadlock

Deadlock in this study is defined as the situation where black cells exist on the board but cannot make a movement while they have seized a white cell using the search algorithm.

3.2 Retrieval Scenarios

Different scenarios for retrieving items from the board were considered for this study. Some of the scenarios were more prone to deadlocks resulting in decreased performance of the puzzle. Some scenarios were more difficult to implement while maintaining agent-based logic. The following are some of the scenarios that are examined:

3.2.1. Assigning white cells to specific areas of the puzzle

In this scenario, the puzzle is divided into subareas, as shown in Figure 1. One or more white cells are assigned to each subarea. White cells can only move within the specified subarea. It was found that this policy lowers the performance of the puzzle since white cells are prohibited from serving the requested item outside their subarea even if it happens to be next to the white cell. As another example, if an item is requested for retrieval outside its current subarea, the policy requires a white cell from the items storage subarea as well as a white cell from the retrieval subarea to exit the puzzle. This policy impedes optimal use of the white cells and makes the implementation very complex.

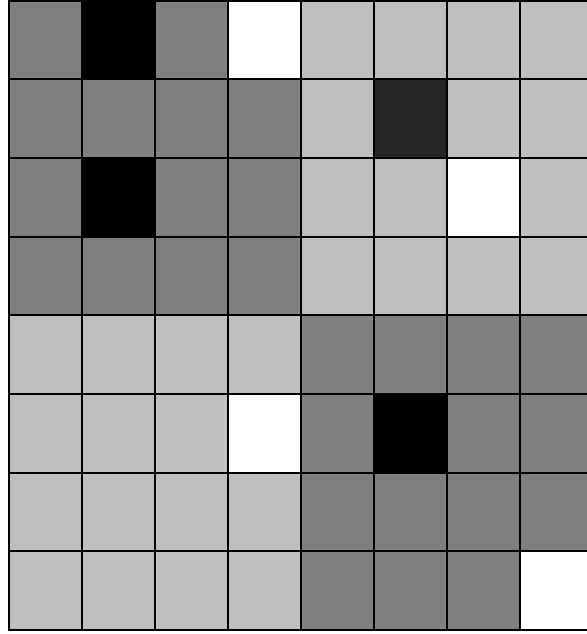


Figure 1: Designed puzzle with assigned white cells to the specific subareas of the board

3.2.2. Requiring at least one empty cell in each column and row

Gue et al. (2014) used an algorithm that forced each column to keep at least one empty cell to avoid deadlock. In this algorithm, empty cells move to open a temporary aisle in the board. This was not implemented because it obviously lowers the storage density of the system thus negatively affecting its performance.

3.2.3. Assigning a white cell to a requested item through entire retrieval

After a white cell is assigned to a requested item, it serves only that requested tote until it is retrieved. After the tote is removed from the board, the white cell can be assigned to a different item. This scenario lowers the complexity of programming. Empty cells can be assigned on a first come first served (FCFS) basis. However, it results in a higher average number of movements since it ignores white cells that may be closer once a move is made.

3.2.4. White cells assigned based on nearest distance after every movement

The nearest white cell is a cell that makes the minimum number of movements to get to the optimum side of the requested item. The optimum side is the side that has the least distance from the retrieval point. When a message gets to the nearest white cell based on a message passing protocol, the empty cell will be reserved to serve the requested item. This cell is only reserved until the board makes a move. Technically, a board move is the white cell movement from a cell on the board to another cell, as shown in Figure 2.

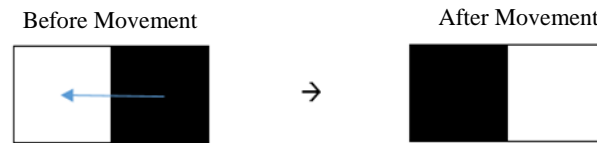


Figure 2: One movement in the board

3.3. Methodology Definitions and Assumptions

Square puzzle was selected for methodology implementation. Each cell in the board is an agent and can store one item. Similar algorithms are incorporated in all cells. Stored items are shown as gray. Requested items are shown in black. When requested, the cell changes its color from gray to black. Empty cells (escorts) are white.

As mentioned previously, the study intended to solve a puzzle where an item could be retrieved from any point of the puzzle border. As noted in section 2.3, many researchers have reported that puzzle based networks are PSPACE-Hard and do not have a mathematical solution when the puzzle dimensions grow. Therefore, simulation is the viable approach for solving the

problem. The following configurations were used in this study for simulating the puzzle problems:

- 1) The simulated puzzle dimensions were 3×3 , 6×6 , 12×12 , 20×20 , 25×25 and 50×50 .
- 2) The number of black cells was always 3 which means 3 items were concurrently requested from the board.
- 3) The number of white cells varied from 1 to at least half the number of puzzle cells.
- 4) Retrieval points are randomly selected from the puzzle border.
- 5) Then, 3 cells within the puzzle are randomly selected for retrieval. Their color changes from gray to black.

In previous research studies, items could be moved only in north-south directions for replenishing or retrieving purposes, or individual conveyors could use message-passing protocols to provide an empty spot in east-west directions. The algorithm used in this study for solving the puzzle has two different phases; search and movement. In the first phase, escort cells are assigned to the requested items. If the number of requested items is larger than the number of escorts, then requested items compete to seize an escort (empty spot). If the number of escorts is larger than the number of requested items, then closest escorts to black cells will be selected according to the search algorithm. Then the puzzle enters the moving phase where escorts make a move. This cycle of search and move phases continues until all requested items are retrieved and the puzzle is solved.

-Network Search and Move Phases

One-time step of the network includes both search and moving phases. The network goes through these phases in as many time steps as required to retrieve all requested items and solve the puzzle. This is shown schematically in Figure 3.

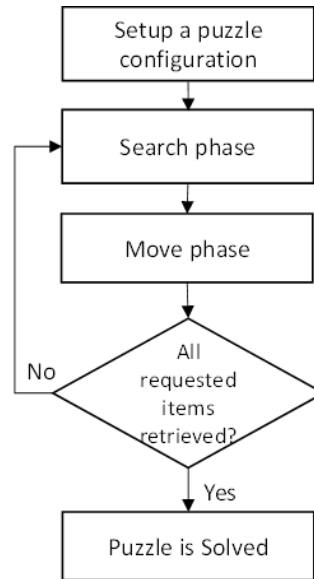


Figure 3 Overall steps in solving a puzzle.

To further explain the cycle illustrated above, it is necessary to define a few definitions and illustrate the algorithm developed for this study.

- Optimal Side

Optimal side is the nearest side of a requested cell to the retrieval point. Selecting the optimal side is an important step toward making an optimal movement in the puzzle. A simple example is shown in Figure 4. Cell 1 borders the optimal side of the black cell. Since the item is requested from the left side of the puzzle, white cells compete to serve the requested item from that side. As a result, the white cell closest to the optimum side of the requested item moves to reach cell number one shown in the figure. It is apparent that the cell at the bottom of the puzzle is closer to the optimal side and will be seized by the black cell. If two sides of the black cell

qualify as optimal sides, the algorithm selects the side that first calculates seizure of which white cell results in optimum for retrieval. The white cell that can retrieve the item with the most number of 3-moves is selected.

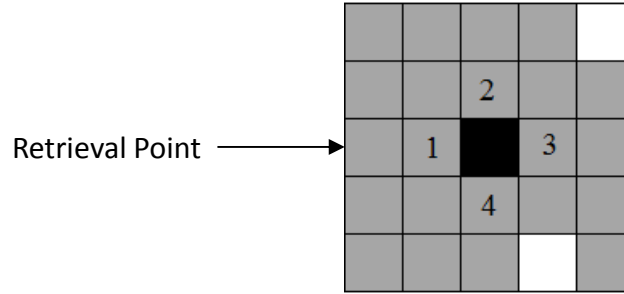


Figure 4: Optimal side of requested item identification

-White Cell Seizure

Black cells seize white cells to move. As was explained earlier, the seized white cells moves to arrive to the optimal side of the black cell. When there is more than one white cell available to the black cell, the closest one to the optimal side is seized. This is identified based on the agent-based algorithm developed. The requested item sends a message to all four directions to seize an empty spot considering its optimal side. If there is no empty spot near the requested item, each of the cells receiving the message sends it to three different directions available to search for an empty spot. This message passing continues until an empty spot is seized. The black cell is informed of the seizure through a recursive message passing protocol. After this point, this empty spot is allocated to the requested item that is being retrieved until a move is made. Figure 5 shows an example of how the message-passing algorithm works for finding an empty spot.

Given the retrieval point shown in the figure, the white cell at the bottom of the puzzle is seized for the black cell.

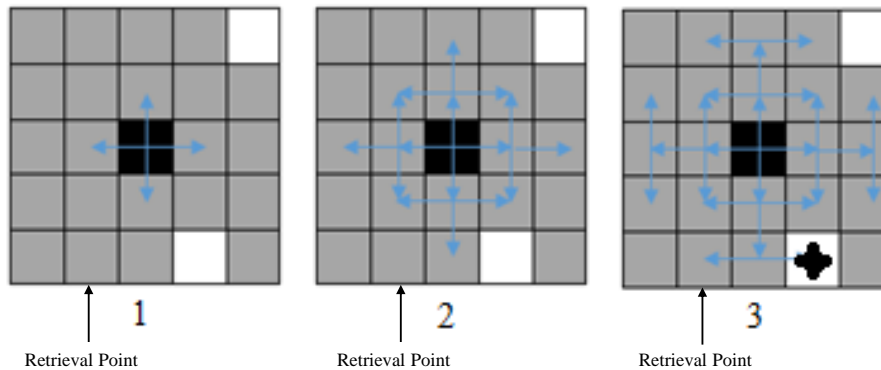


Figure 5: Searching steps for seizing an empty cell

-Optimal Movement

The optimal movement consists of two parts. First, the algorithm tries to move the seized white cell into the optimal side, and then starts making 3-moves and 5-moves, as are explained by Gue and Kim (2007). Figure 6 demonstrates steps involved in 3-moves scenario starting with the puzzle numbered “1” in the top left-hand corner as the current position of the board before any movement. Figure 7 represents a 5-move scenario that similarly starts with puzzle numbered 1 at the top left hand corner.

Gue and Kim (2007) devised a theory that rationalized the 3-moves and 5-moves scenarios in a puzzle. They proved that the optimal number of movements in a puzzle occurs when the black cell makes as many 3-moves as possible until retrieval. There are circumstances that the only available option to make further movement toward retrieval point is 5-moves.

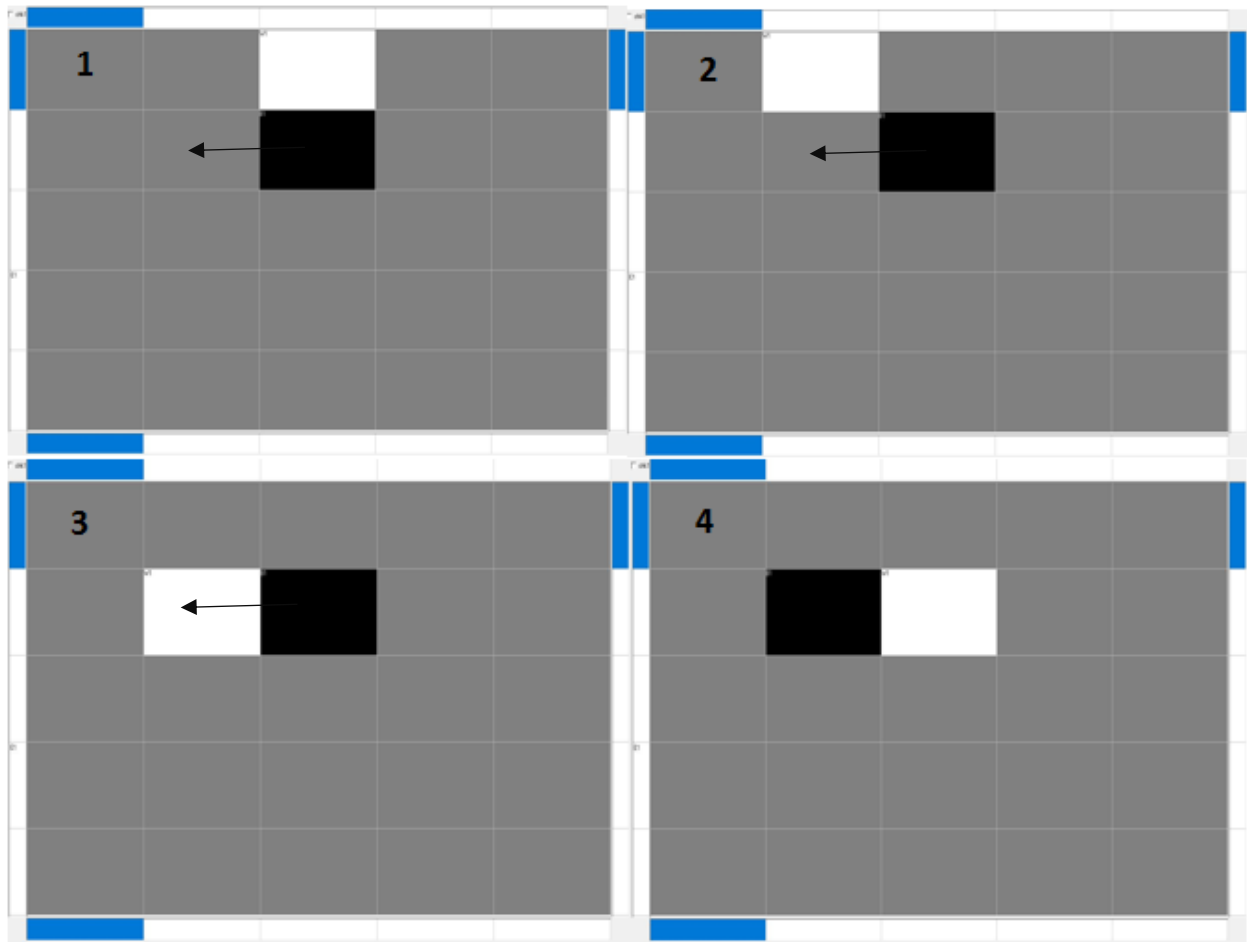


Figure 6: Schematic of 3-Moves Scenario

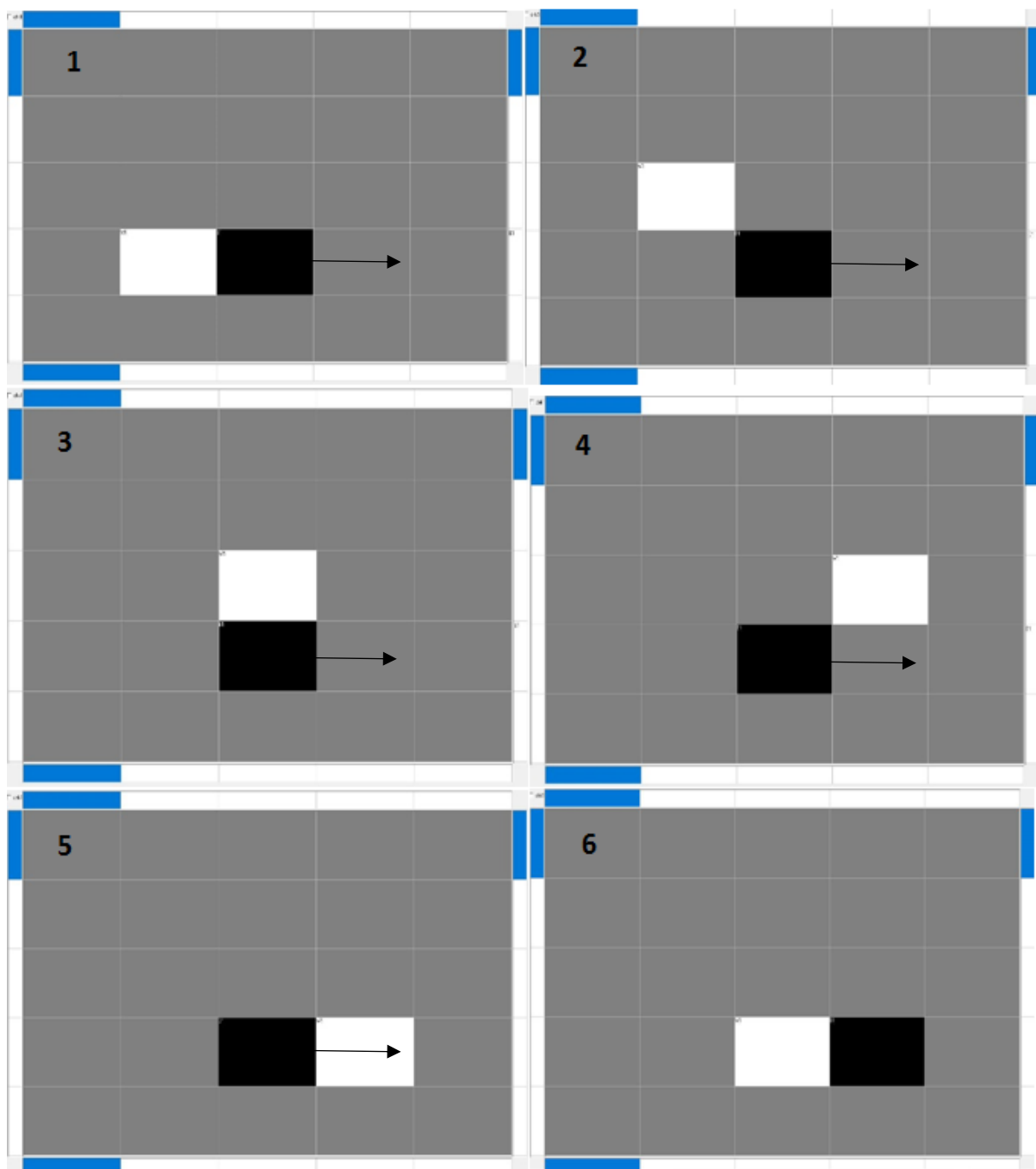


Figure 7: Schematic of 5-Moves Scenario

-Deadlock Prevention

Deadlock is the situation where a black cell seizes a white cell but they cannot make a movement. There are a number of situations that may result in deadlock. Sometimes deadlock occurs because more than one black cells have seized the same white cell, as shown in Figure 8. Another situation that may create a deadlock is when one black item seizes two or more white cells with the same distance to the optimum side of the black cell, as shown in Figure 9. Deadlock may also occur even when black cells have seized different white cells but prevent a movement as shown in Figure 10.

During the simulation, these deadlock instances required additional agent-based logic. To detect a deadlock situation, additional function was incorporated in every agent's logic to detect if a requested item with a seized white cell does not move. This triggers the deadlock prevention algorithm. For the first and third examples of deadlock illustrated in Figures 8 and 10, the algorithm temporarily terminates the demand for both black cells essentially making them gray and randomly picks a number from zero to 3. This random number indicates the number of moves that each cell remains gray. If both cells happen to randomly choose the same number, they both turn black at the same time, which again creates the deadlock and triggers the deadlock prevention algorithm. This continues until the board exits the deadlock situation.

In the second deadlock case shown in Figure 9, both white cells inform the black cell through recursive function that they are seized for the black cell. Black cell has seized the white cells and cannot move which is indicative of a deadlock situation. The deadlock prevention algorithm randomly selects one of the white cells.

Other potential deadlock situations can be conceived such as permutation of the situations illustrated. The algorithm is capable to exit the deadlock situation without the interference and with agent-based principals.

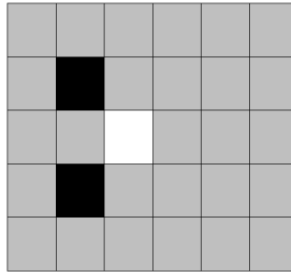


Figure 8: First deadlock example: competing black cells to seize the same white cell at equal distance.

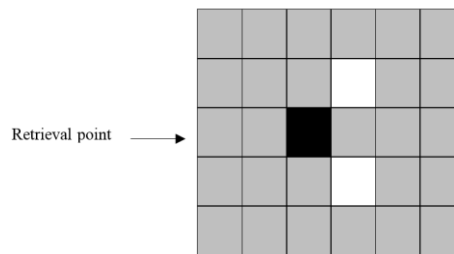


Figure 9: Second deadlock example: black cell seizing 2 white cells at equal distance from the optimal side.

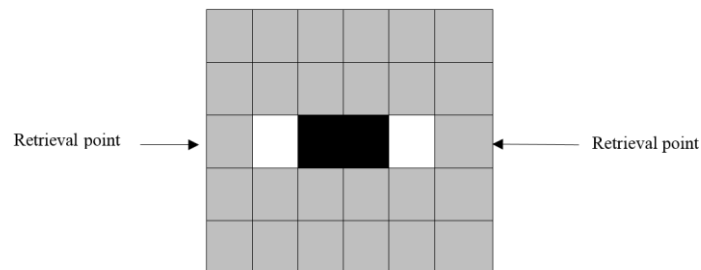


Figure 10: Third deadlock example: Colliding black cells being retrieved from opposing sides while seizing different white cells.

3.4. Methodology Flowcharts

The algorithm for solving the puzzle network is illustrated in Figure 11. As shown, the solution starts with identifying black cells in the puzzle, finding an optimal side for retrieval, seizing nearest white cell and resolving potential deadlocks. These are the component of the network in the search phase. Then, the puzzle enters the moving phase. In this phase, either a white cell moves closer to the black cell, or the black cell moves toward the white cell. If the black cell is at retrieval point, the item is retrieved and the black cell turns white.

Figure 12 presents the algorithm for resolving the deadlocks. As was discussed previously, three simple scenarios for deadlock were deliberated and discussed. Algorithms were developed for resolving those scenarios. Since situations that are more complex may encompass a combination of these deadlock scenarios, the algorithm steps into a cycle to assure even the most complex deadlock situations are resolved using the agent-based logic.

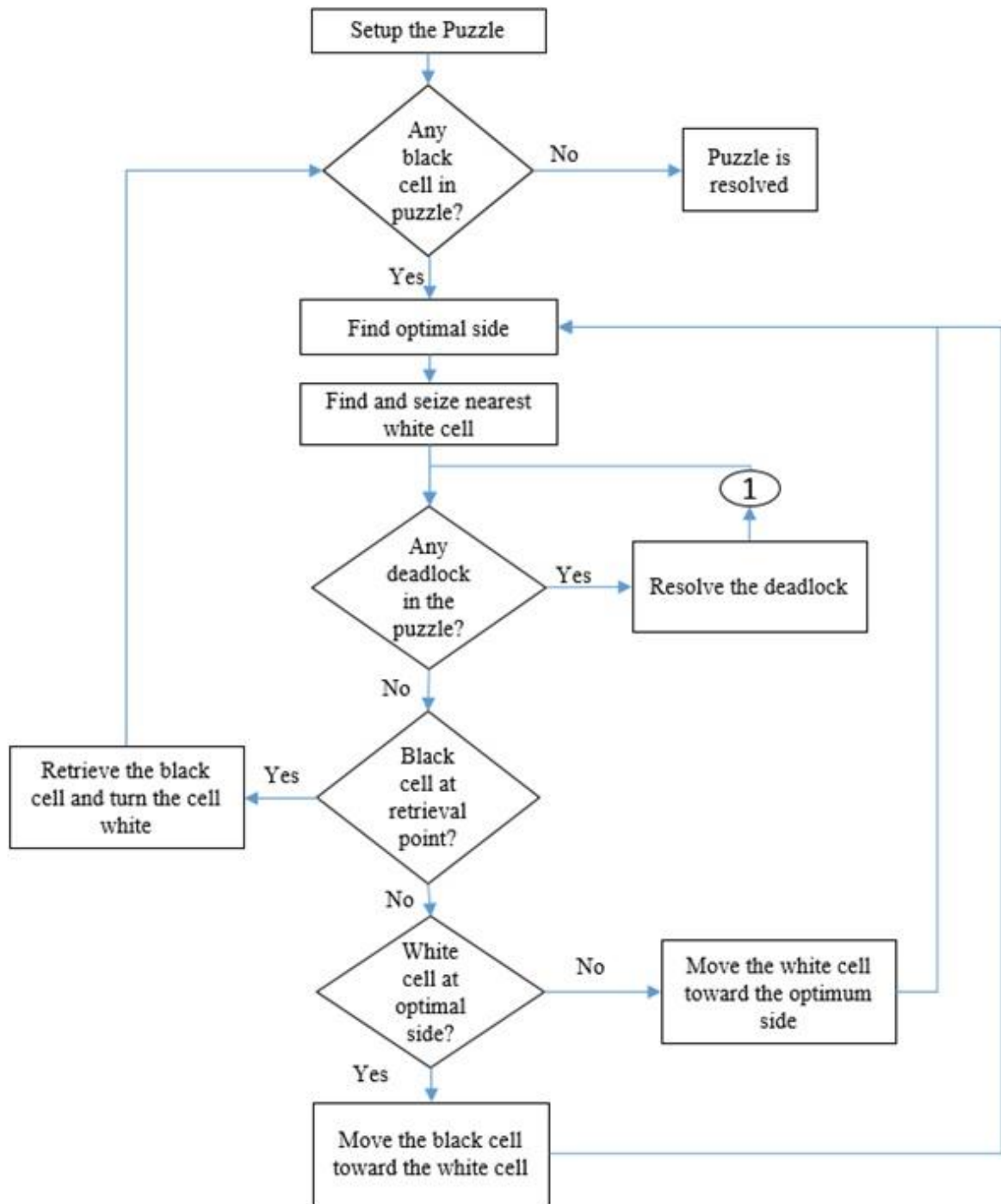


Figure 11- Overall puzzle network algorithm

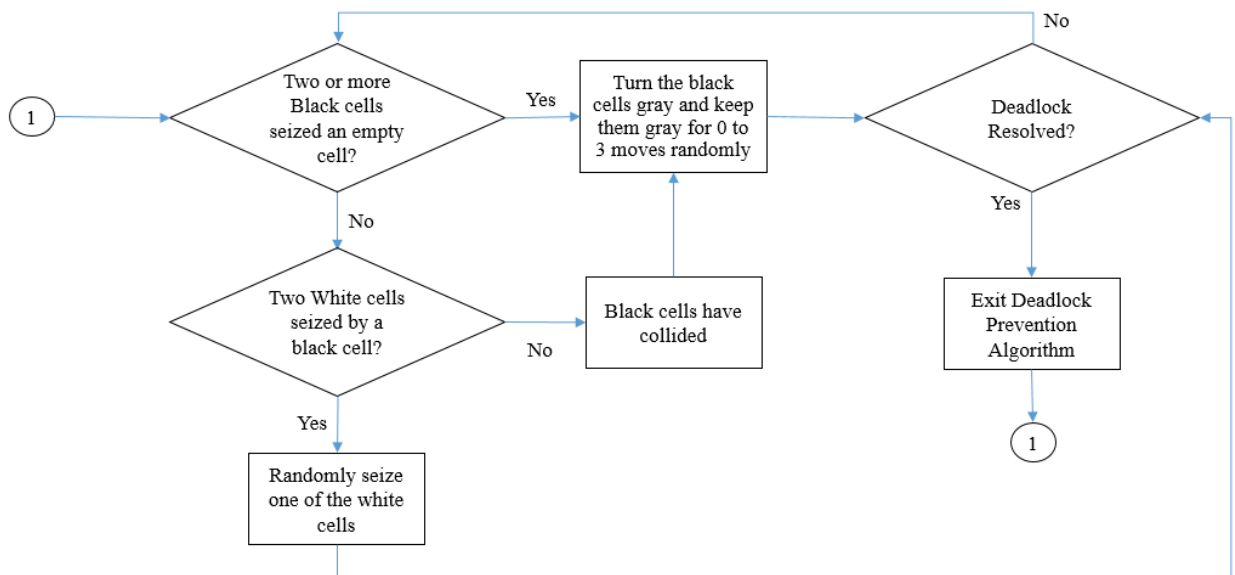


Figure 12- Overview of deadlock prevention algorithm

3.5. Puzzle Network Retrieval Examples

Two examples are illustrated in this section as follow.

-Puzzle Retrieval Illustration: Example One

The following presents an example of a network puzzle with a black cell and two empty cells. The black cell should be retrieved from the top right-hand corner of the puzzle. Figures 13 and 14 presents snapshots of the steps from the software program. The existing puzzle is illustrated on the frame number #1. The optimal side of the requested item is marked in red. The black cell searches for the nearest escort (white cell). Both white cells have the same distance to the optimal side of the black cell. This is one of the deadlock cases. Using the embedded deadlock prevention algorithm, the black cell seizes one of the white cells randomly. The white cell at the bottom was selected and moved to the right closer to the optimal side, as shown in the second frame. Next, the same white cell is seized since it is now closer to the optimal side than the other white cell. It makes another move toward the optimal side, as shown in the third frame.

This continues until the white cell arrives to the optimal side shown in the fifth frame. Then, the black cell makes a move upward as shown in the sixth frame. After the black cell moves to the position of the escort, or empty cell, the next white cell should be seized based on its proximity to the optimal side of the black cell. This time, the white cell at the top of the board is closer to the optimal side and will be seized and making the movement toward the optimal side as shown in frame 7 through 9. In frame #9, the white cell is on the optimal side of the black cell, therefore the black cell makes an upward movement shown in frame 10. The cycle of searching for the closest white cell and the associated movements continue as shown in the remaining frames until the item exits the board. A white cell replaces the black cell in the last frame.

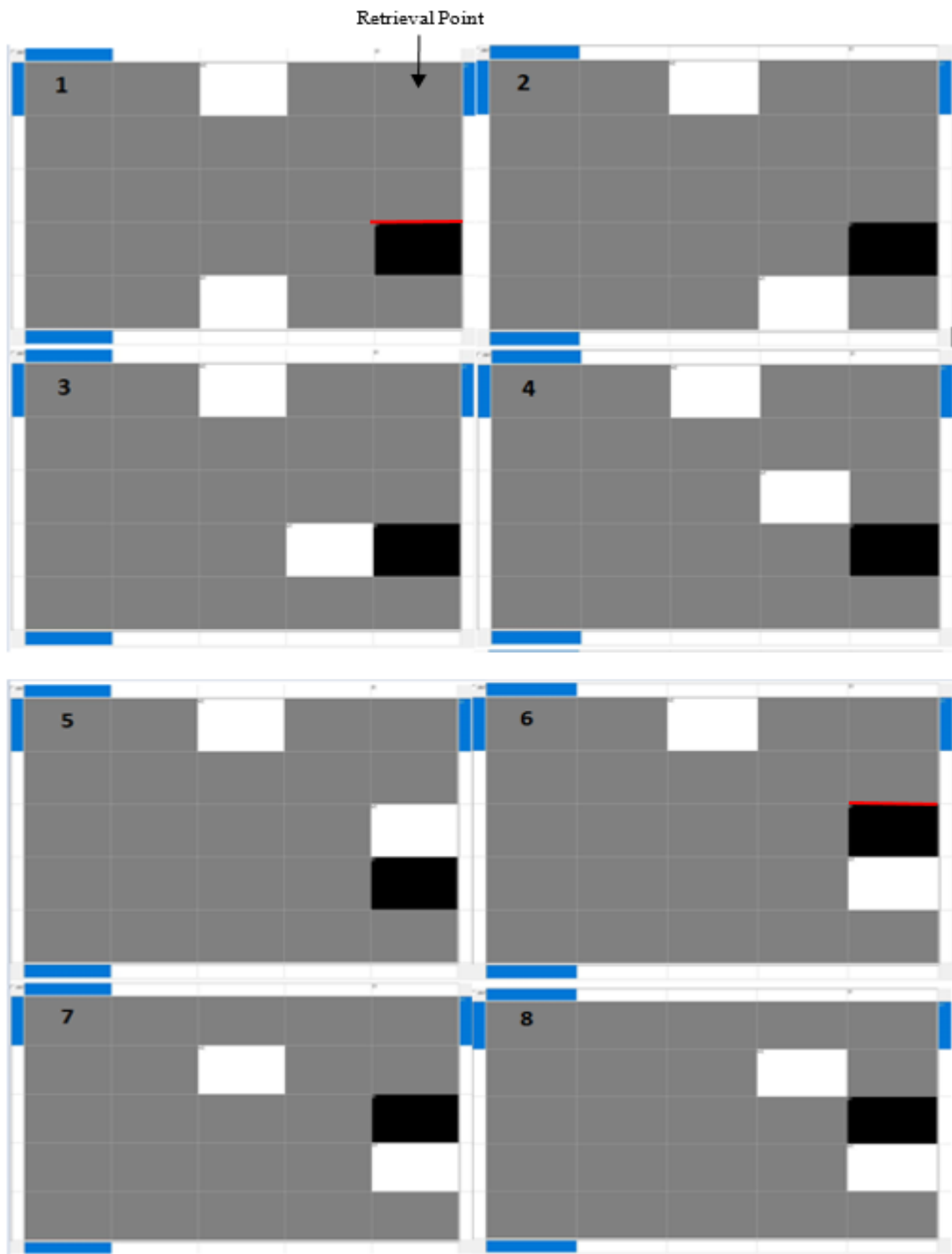


Figure 13: illustration of how puzzle works

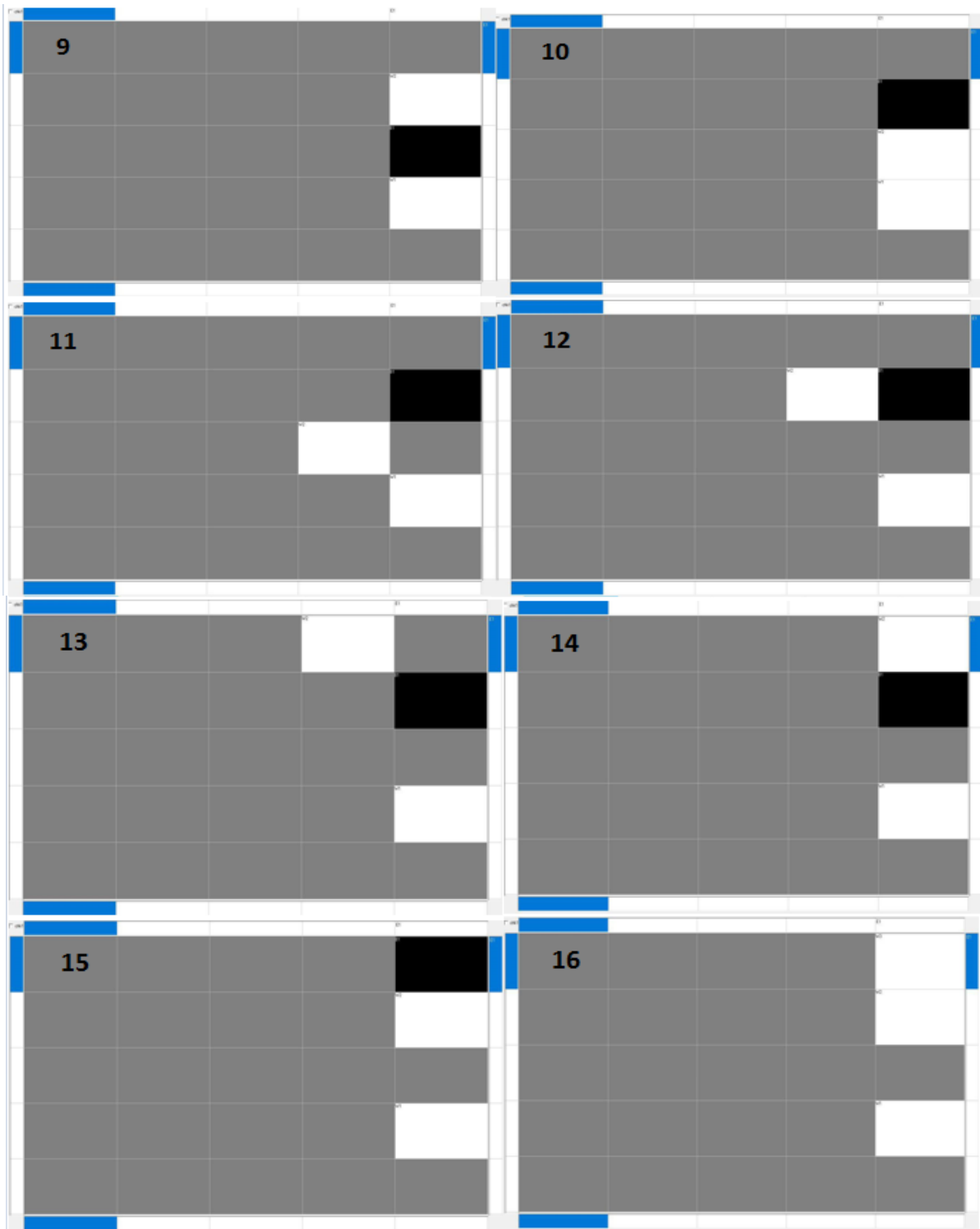


Figure 14: illustration of how puzzle works

-Puzzle Retrieval Illustration: Example Two

This example presents an interesting case where there are 2 optimum sides and 2 white cells each serving one of the sides. As shown in Figure 15, the puzzle includes one black cell and 5 white cells. The retrieval point is at cell number 3. Two sides of the black cell, marked in red, qualify as the optimal side since they have the same distance from the retrieval point. Then, the algorithm searches to seize the nearest white cells for each of the optimal sides. If the nearest white cell to one of the sides is closer than the other one, the algorithm seizes that one for the move. In the illustrated example, the nearest white cell of both optimal sides have the same distance from their respective sides. In this case, the algorithm seizes the white cell that can retrieve the item using the most number of 3-moves.

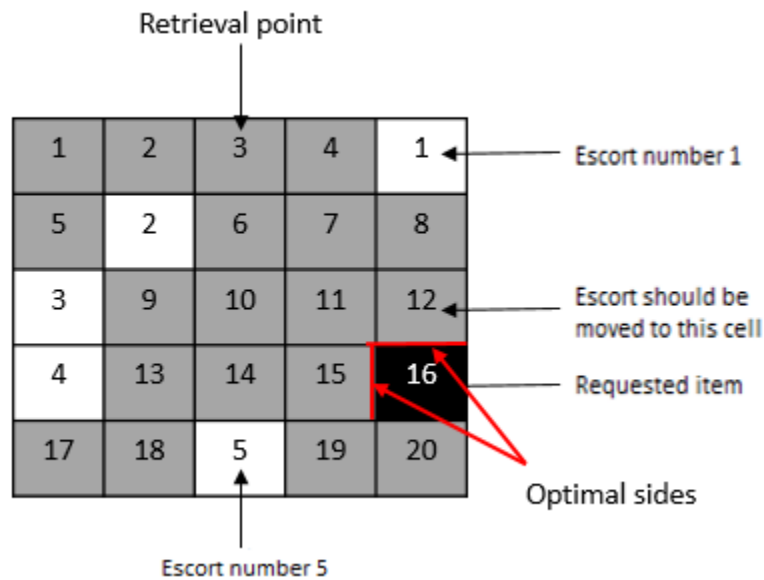


Figure 15: Choosing an escort and retrieval path

Figure 16 shows five possible paths that can retrieve the requested item from the puzzle. As is evident from the figure, the (d) path uses the most number of 3-moves for retrieval. Therefore, the algorithm seizes escort number 1 and retrieves the item using the (d) path.

Table 1: Associated number of movements based on selected escort and path for example 2.

	(a)	(b)	(c)	(d)	(e)
Number of movements with escort number 1	23	21	19	15	19
Number of movements with escort number 5	21	19	21	17	17

CHAPTER 4. ANALYSIS AND RESULTS

4.1 Simulation Results

The object-oriented program developed in this study was used to simulate more than 50,000 iterations representing a variety of puzzle configurations. The number of movements was used as a surrogate variable for request items retrieval time. It is accepted that as the number of movements increase, so does the retrieval time. The simulation intended to measure the impact of number of empty cells and puzzle size on the number of movements for retrieving requested items.

Figure 17 illustrates the steps involved in calculating the simulation results. As shown, a puzzle configuration with specific size and number of white cells is first generated. Then, the white and black cells are randomly placed in the puzzle. Also, a random retrieval point is assigned to every requested item. The number of movements for retrieving every one of the three requested items are recorded and averaged for the first iteration. This average is recorded as the result for the first iteration. Then, for the next iteration, the location of the white and black cells are randomized again. This cycle continues for 100 iterations. Then, the recorded number of movements from all iterations ($M_{i, ave}$) are averaged and recorded as the result for the puzzle configuration with the chosen size and the number of white cells. This is shown as Average Retrieval Movement (ARM) in the figure.

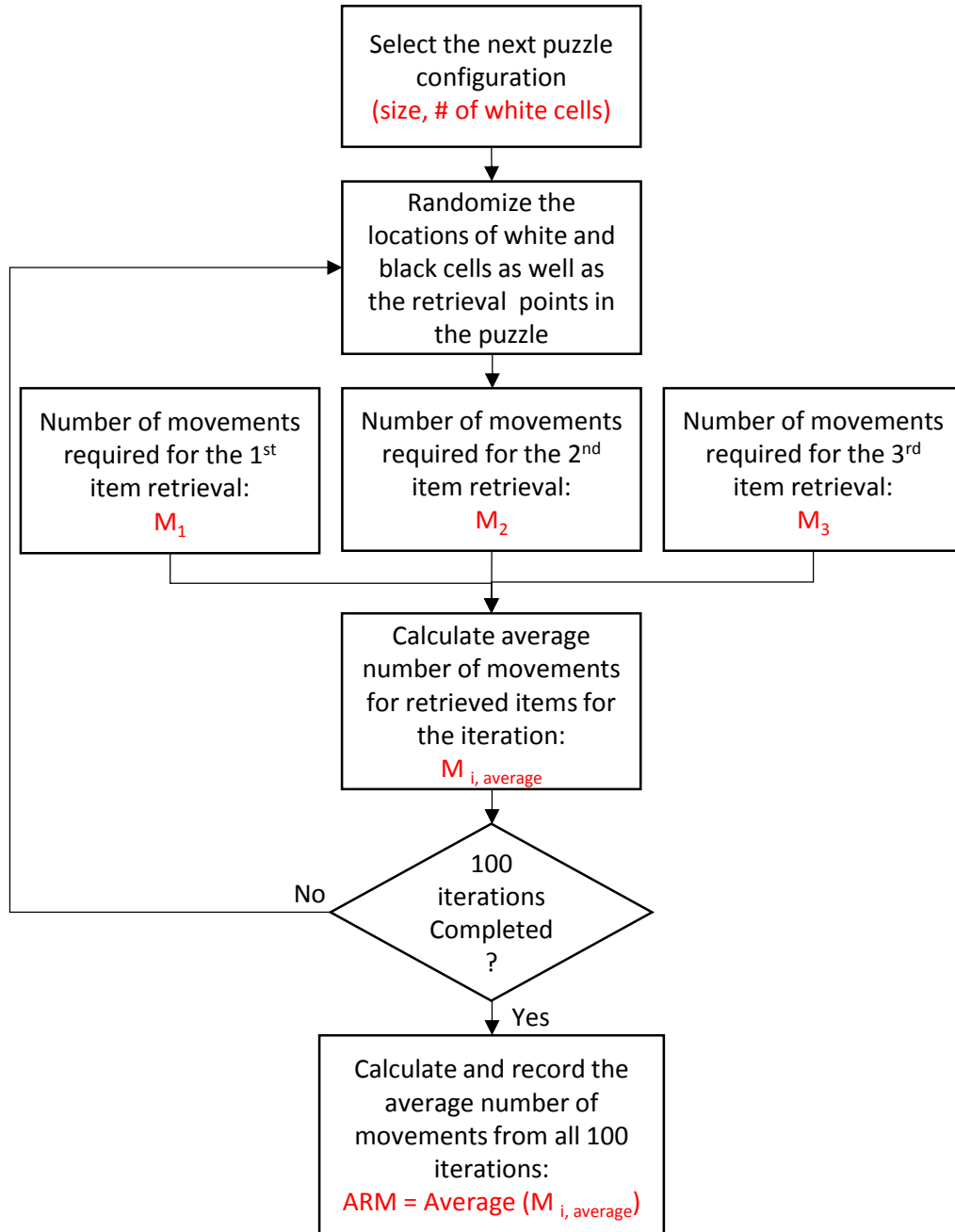


Figure 17: Steps in Calculating Simulation Results.

Figure 18 presents the iteration results for the calculated average number of movements for retrieved items ($M_{i, ave}$) for a sample puzzle setup. In this example illustration, the puzzle size was 6×6 and the number of white cells was 18. There is a relatively high variation in the average

number of movements to retrieve the 3 requested items. In a randomized configuration, when the white cells happen to be located close to the black cells, and the black cells happen to be located close to their associated retrieval points, the average number of movements to retrieve the items is expectedly low, and vice versa. As shown in Figure 18, the average number of movements in the 34th iteration is rounded to 4. This represents the minimum average number of movements for this puzzle configuration. On the other extreme, the average number of movements for retrieving the 3 requested items in 37th iteration is shown as 25. In the remaining iterations, the average number of movements fluctuates between these 2 extreme values. The average from all iterations, named average retrieval movements, is the simulation result for this puzzle configuration and is calculated to be 10.4.

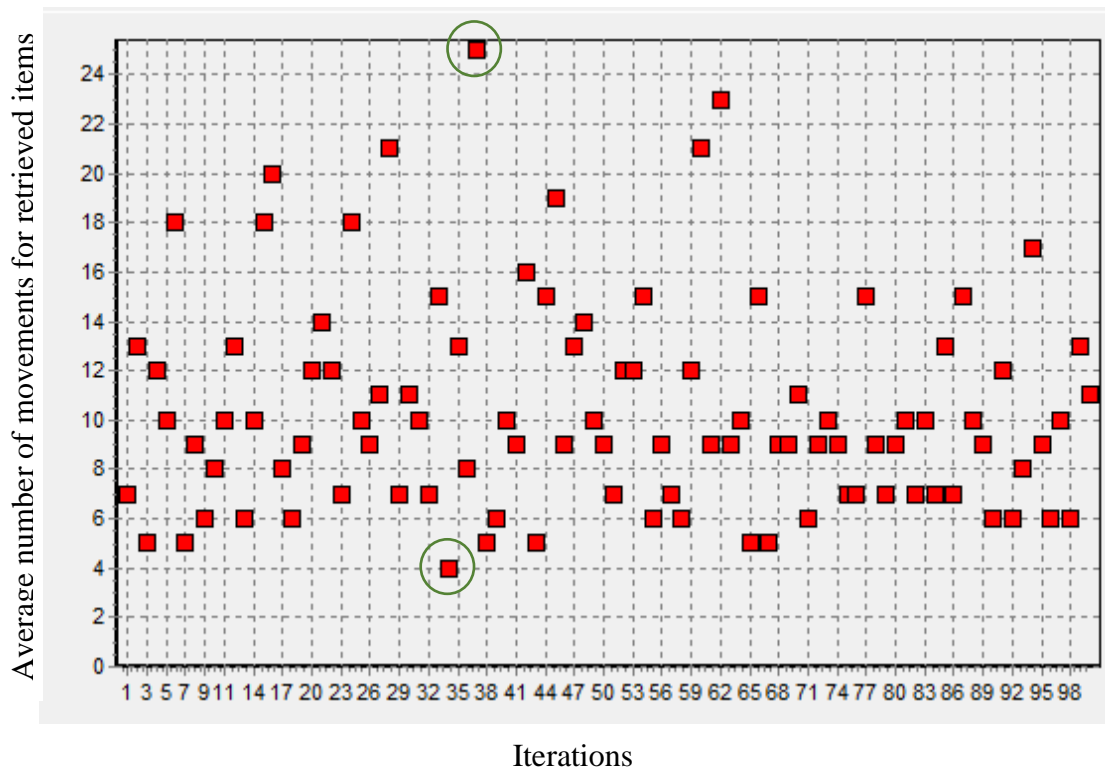


Figure 18: Average number of movements for retrieved items in every iteration for 6*6 puzzle with 18 white cells.

Figure 19 presents the simulation results for 6×6 puzzle with 3 concurrent requested items and 1 through 30 white cells. The labels of the x-axis illustrates the simulation results for puzzle configurations presented by the puzzle size (*a*), the number of requested items (*b*), and the number of white cells (*c*) separated by commas. The y-axis presents the average number of movements required for retrieving the requested items. It should be emphasized that the number of requested items in this study was 3 for all simulations. Also, the simulation was terminated after all requested items were retrieved from the puzzle.

Every point on this figure represents findings from 100 iterations associated with one puzzle configuration. The simulation results for a 6×6 puzzle with 18 white cells is presented with a red square in Figure 19. This point is the average of the 100 iterations that are shown in Figure 18.

As shown in Figure 19, when the number of white cells increases in the 6×6 puzzle from 1 to 3, the average retrieval movements (ARM) slightly increase. Then, as the number of white cells increase further from 3 to 25, the ARM constantly decreases. Adding white cells beyond 25 does not have as much impact on reducing the ARM.

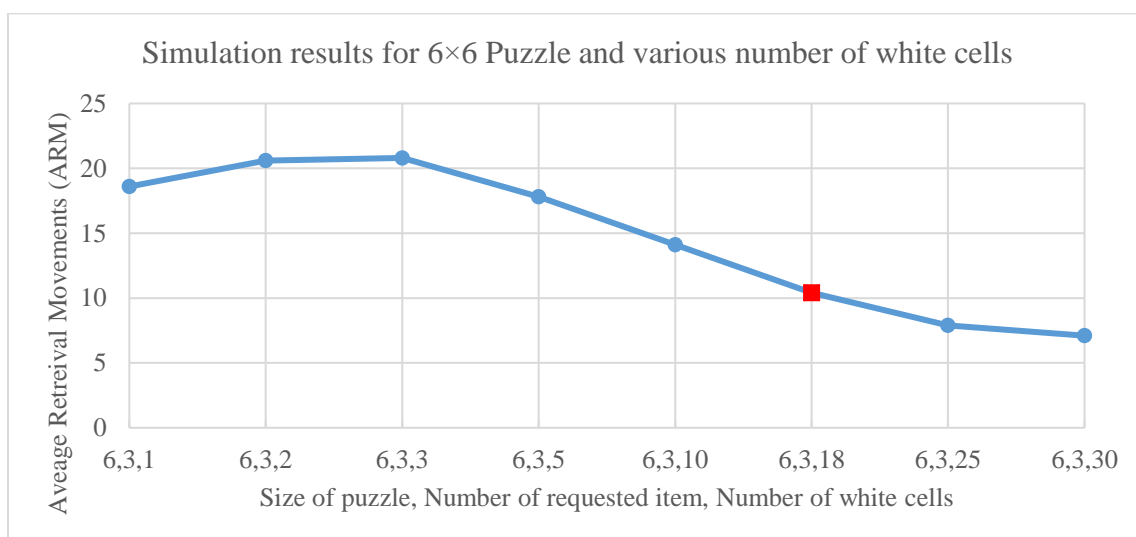


Figure 19- Simulation results for 6×6 puzzle

Figures 20 through 25 illustrate results for various puzzle configurations. Figure 20 depicts the average number of movements for 100 iterations for a 12×12 puzzle with 18 white cells. Figure 21 presents the simulation results for a diverse 12×12 puzzle setup with varying number of white cells from 1 to 100. The sharpest decrease in ARM occurs when the number of white cells are between 10 and 62. The figure plateaus when the number of white cells passes 90.

Figures 22 and 23 illustrate the findings for a 20×20 puzzle. Figure 22 presents the average number of movements for retrieved items when 18 white cells are used in the puzzle setup. The number of movements varies from a minimum of 41 to a maximum of 142 among the iterations. According to Figure 23, in some of the intervals for the number of white cells such as 1 to 3 and 10 to 40, the ARM does not change much. The puzzle experienced the sharpest decrease in ARM when number of white cells decreased from 40 to 100.

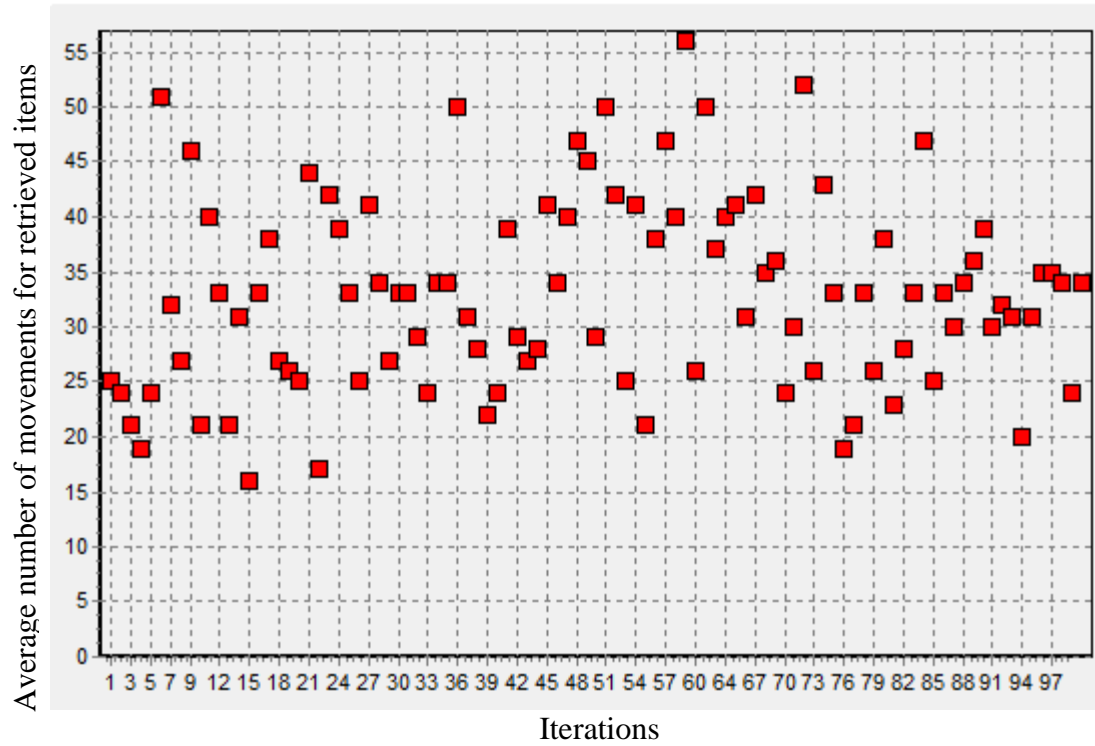


Figure 20: Average number of movements for retrieved items in every iteration for 12×12 puzzle with 18 white cells.

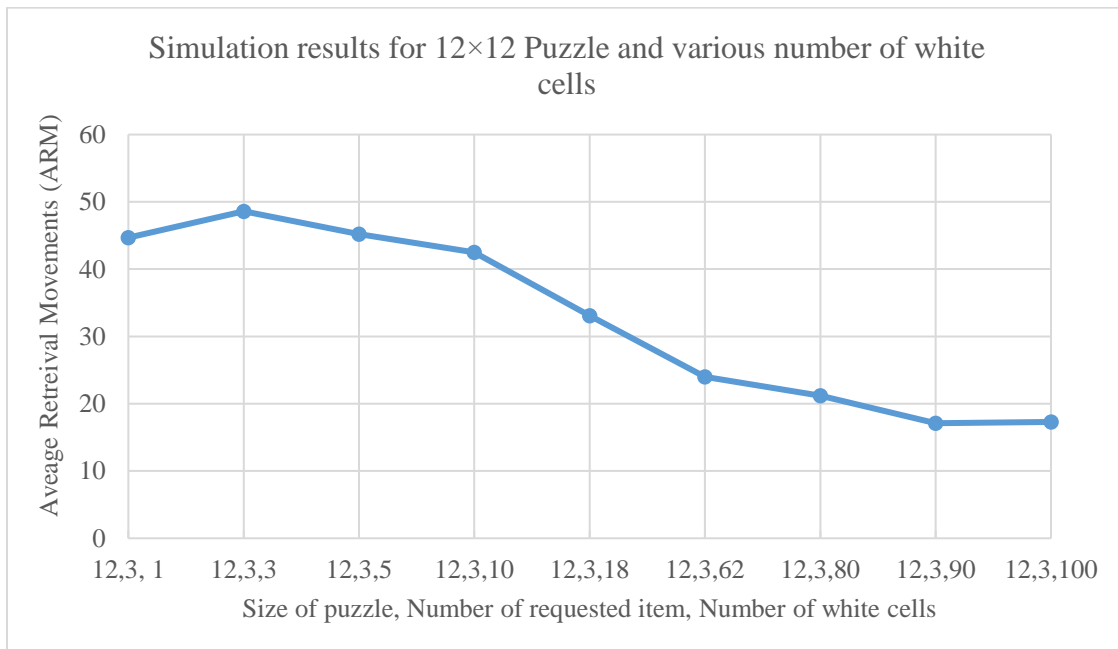


Figure 21: Simulation results for 12×12 puzzle

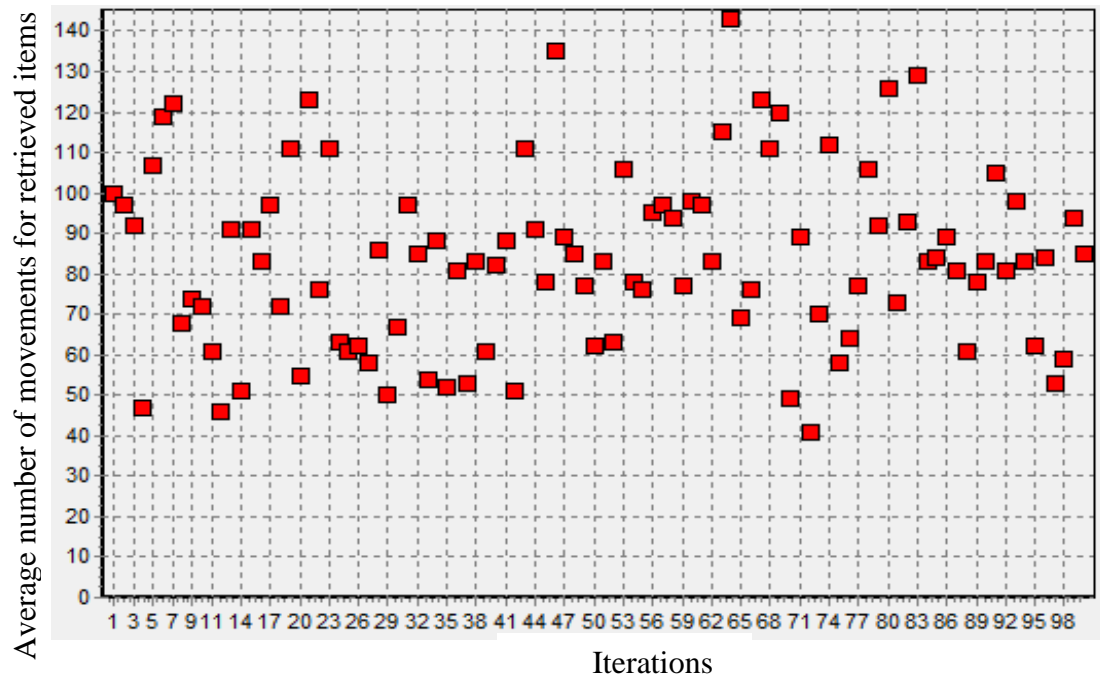


Figure 22: Average number of movements for retrieved items in every iteration for 20×20 puzzle with 18 white cells.

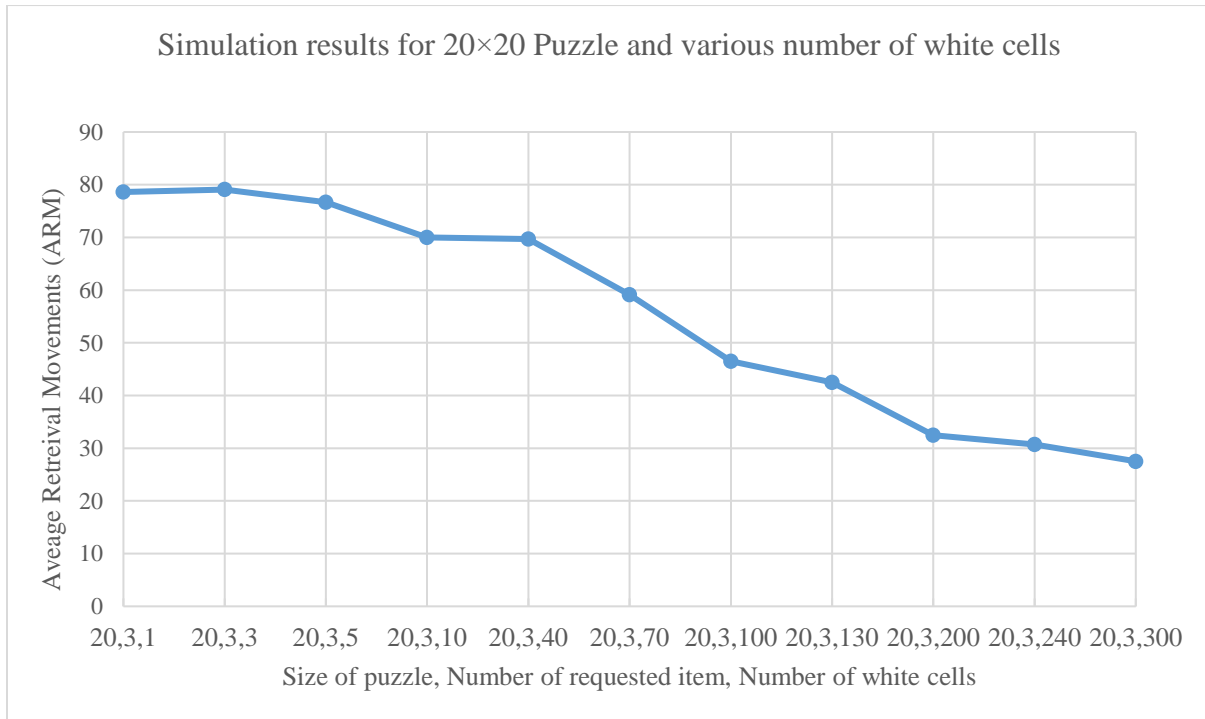


Figure 23: Simulation results for 20×20 puzzle

Findings for 25×25 puzzle configurations are shown in Figures 24 and 25. As shown in Figure 24, the minimum average number of movements for retrieving the requested items was 33 and its maximum was 149 when the number of white cells was 18. As evident from Figure 25, the deepest decline in ARM was observed when there were 40 to 150 white cells. Also, it is shown that increasing the number of white cells from 1 to 40 reduced the ARM by only about 20% from 100 to 80. The curve almost plateaus for the number of white cells beyond 250.

Based on the simulation results, the average number of movements for retrieving items follows a relatively similar pattern. When the puzzle size is smaller, the curve presents 3 distinct trends. The curve is almost flat in the beginning and then decreases sharply. Toward the tail of the curve, the curve levels again. This is the case for puzzle sizes up to 12×12.

When the puzzle size grows, this pattern repeats itself twice. This means, the curves starts out relatively flat, follows a sharp decrease, levels again, and then the curve drops sharply for a second time and plateaus at the tail of the curve. This pattern is observable in 20×20 and 25×25 puzzle sizes. It is important to note that the puzzle configurations presented in Figures 16 through 23 are only illustrative examples of the many configurations that were simulated by the software.

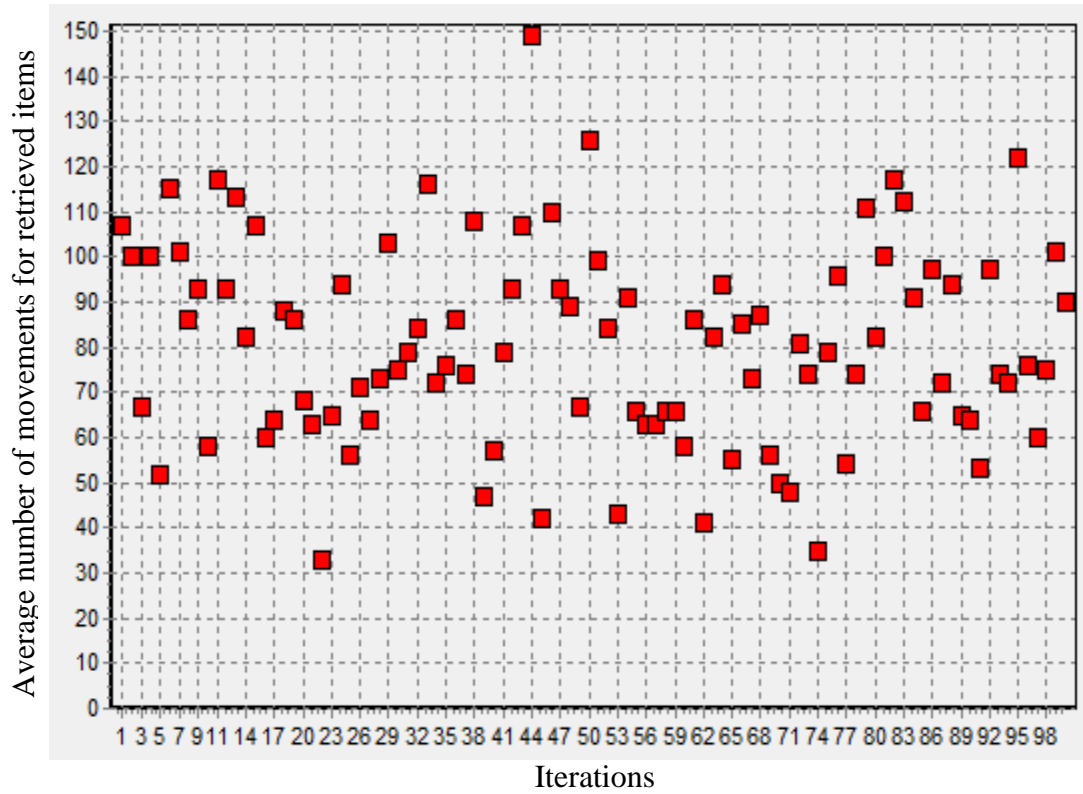


Figure 24: Average number of movements for retrieved items in every iteration for 25×25 puzzle with 18 white cells.

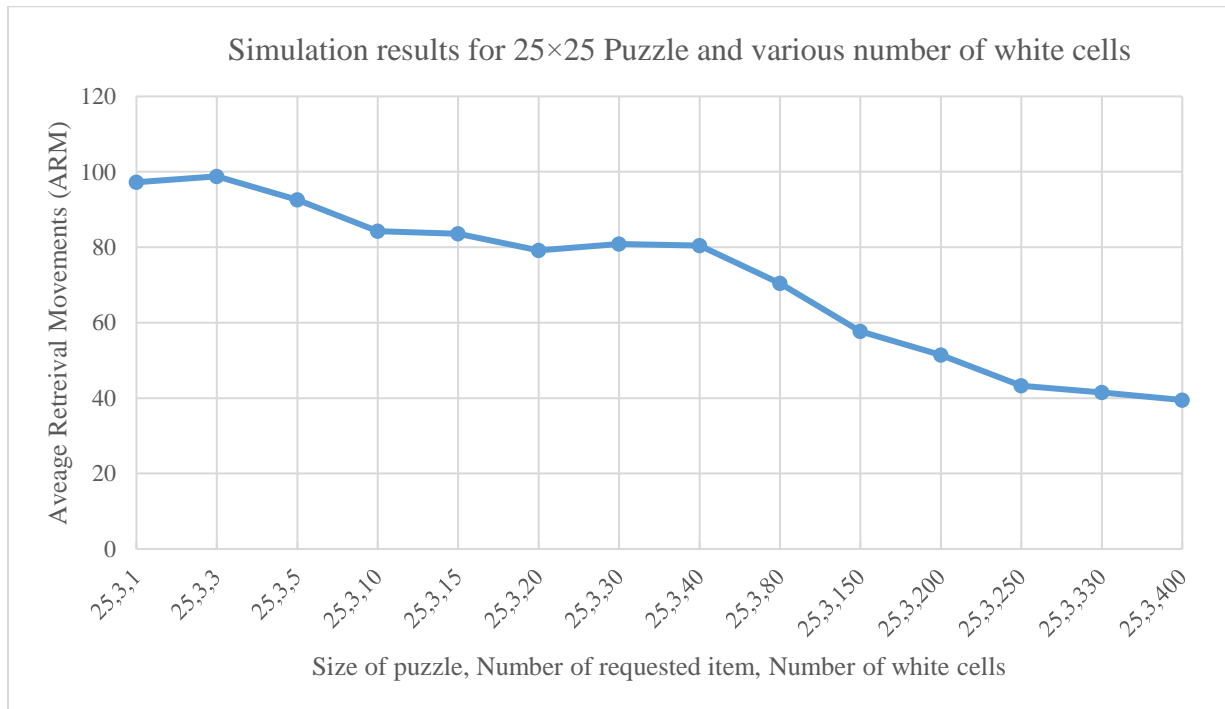


Figure 25: Simulation results for 25×25 puzzle

The impact of puzzle size on ARM was also investigated. Figure 26 depicts the ARM for various puzzle sizes when only one white cell was available in the network. As shown, when the puzzle size increases, the average retrieval movements (ARM) increases as well. The puzzle size almost doubles in every subsequent configuration in the figure, and so does their associated ARM. Figure 27 illustrates a logarithmic relationship between ARM and number of cells in the puzzle.

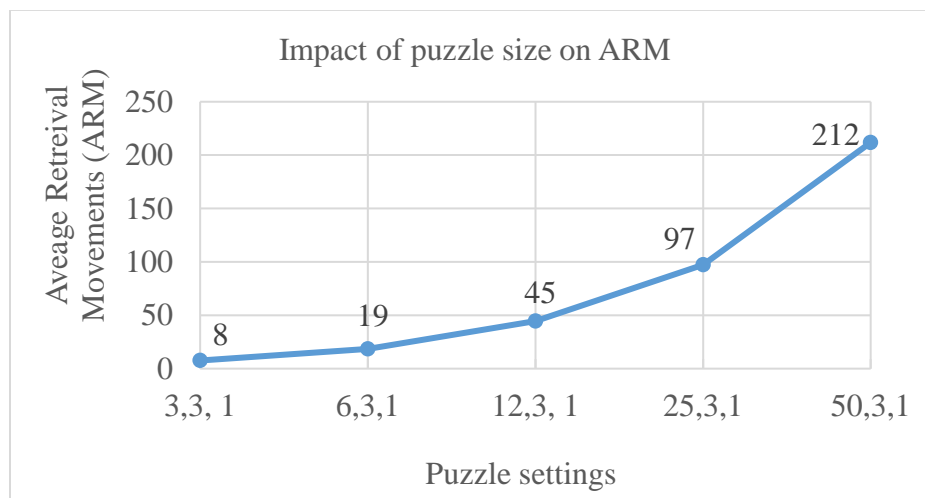


Figure 26: Impact of puzzle size on Average Retrieval Movements (ARM) with one white cell.

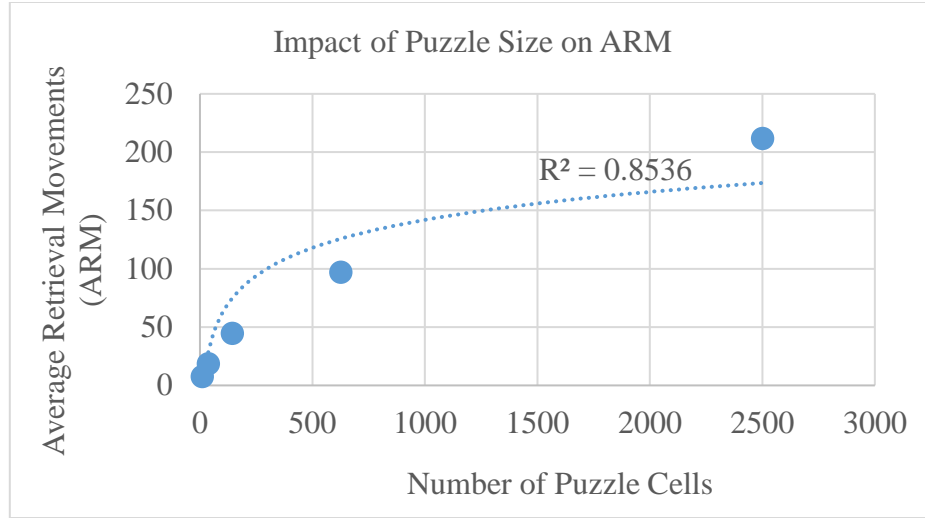


Figure 27: Impact of number of puzzle cells on Average Retrieval Movements (ARM) with one white cell.

4.2 Comparing Simulation Results with Previous Studies

Gue and Kim (2007) mathematically solved the puzzle network for a case with one cell designated for Input/Output in the lower-left corner of the puzzle and with only one item to retrieve and one white cell at the retrieval point. They found that the optimum number of movements for retrieving the requested item is dependent on the location of the black cell with respect to the retrieval point, as shown below:

$$6i+2j-13 \quad \text{when} \quad i > j$$

$$6j+2i-13 \quad \text{when} \quad j > i$$

$$8i-11 \quad \text{when} \quad i=j$$

Where i and j are the row and column of the black cell counted from the retrieval point, as shown schematically in Figure 28 as an example. The i and j of the black cell are 3 and 4, respectively.

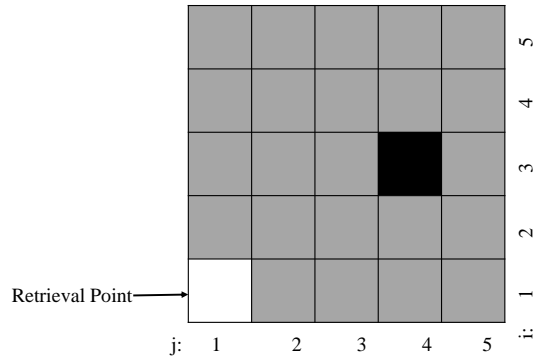


Figure 28- Illustrative example of Gue and Kim (2007) black cell location referencing method.

The software program built for this research was adjusted to simulate the same problem that was solved mathematically by Gue and Kim (2007). This was intended to examine whether the results from the program matches the mathematical optimum solution. Twelve cases were randomly selected as shown in Table 2. In 4 cases, i and j were equal. In another 4 cases, i was bigger than j, and in another 4 cases j was bigger than i. Gue and Kim (2007) relationships were used to calculate the mathematical optimum number of movements for retrieving the black cell. As shown in the table, this equals to the simulation results for all cases.

Table 2: Comparison of software program simulation results with mathematical optimum solution obtained by Gue and Kim (2007).

Case Number	Black cell location (i,j)	Optimum number of movements based on Gue and Kim	Software program simulation results
1	5,5	29	29
2	15,15	109	109
3	25,25	189	189
4	50,50	389	389
5	5,15	87	87
6	15,25	167	167
7	25, 35	247	247
8	35, 50	357	357
9	15, 5	87	87
10	25,15	167	167
11	35,25	247	247
12	35,50	297	297

It is not surprising that the results from the software program match the results from the mathematical formulation developed by Gue and Kim (2007). As was discussed in Section 3.3, Gue and Kim illustrated that for optimal movements, the puzzle should make the maximum number of 3-moves possible for retrieving items. The same study drove the mathematical relationships that were presented above. The validation between results from the previous research and the software program developed in this study is indicative of correct implementation of the algorithms devised by Gue and Kim. Although these cases validate the results for a simplified puzzle, concurrence of the simulation can be used for more complex scenarios simulated in this study since the core of the algorithms are the same.

Taylor and Gue (2008) investigated the influence of empty cell location on the average retrieval time in the puzzle system. This study simulated two of the configurations discussed in Taylor and Gue (2008); random distribution of empty cells and input/output distribution where all the empty cells are at the retrieval point. Figure 29 presents the study results for both configurations. As shown, random distribution considerably outperforms input/output distribution since the average number of movements are smaller for all cases.

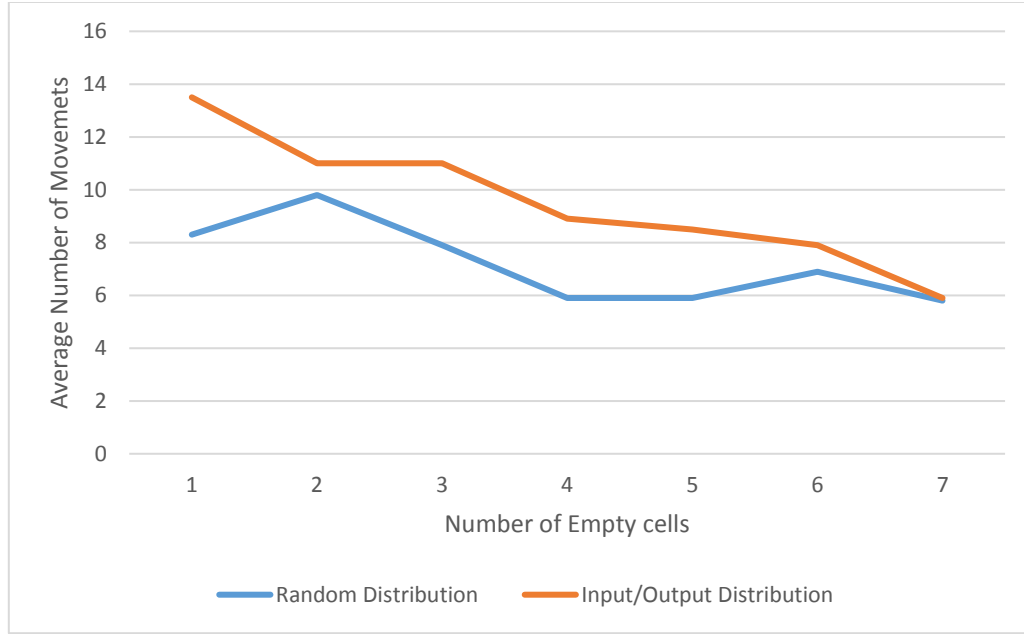


Figure 29: Comparison in average number of movements between Random and Input/Output distributions

Table 3 compares the results from this study with Taylor and Gue (2008). The results from Taylor are estimated from a figure in their paper. In the Input /Output distribution method, results from simulation of this study outperformed results of the previous study as is shown in Figure 30. When comparing the study results from random distribution scenario, they both resulted in relatively similar number of movements as illustrated in Figure 31.

Table 3: Comparison between different configurations of empty spots in Taylor et al. (2008) and current research

		Number of empty spots						
		1	2	3	4	5	6	7
I/O Distribution	Taylor	12	11	10	9	8	8	7
	Current Research	8.3	9.8	7.9	5.9	5.9	6.9	5.8
Random Distribution	Taylor	11.5	11	10	9	8	7.5	7
	Current Research	13.5	11	11	8.9	8.5	7.9	5.9

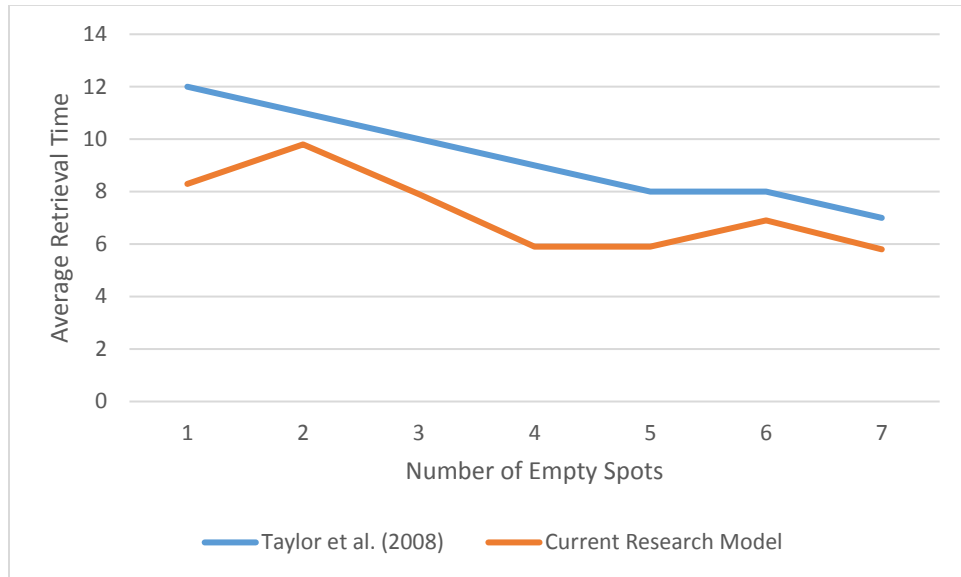


Figure 30: Comparison of I/O distribution

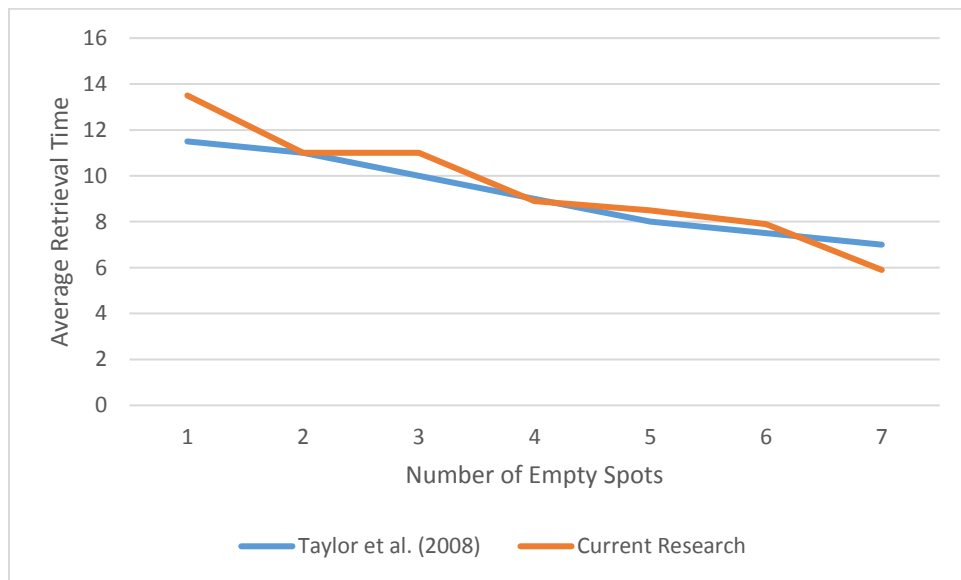


Figure 31: Comparison of random distribution

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 Accomplishments of this research and contributions

Major accomplishments of this research are listed below:

- 1- Developing more than 20 different algorithms to accomplish an agent-based simulation such as a search algorithms, deadlock prevention algorithms and move algorithms.
- 2- Developing a high-density puzzle network model that is able to retrieve totes from sides.
- 3- Developing a high-density puzzle network model that retrieves from one retrieval/replenishment point.
- 4- Comparing results favorably with prior research results of others in the literature.
- 5- Suggesting new high-density systems.

5.2 Conclusion

Based on the observations from the simulations, when the puzzle dimension is small or the number of requested items in the puzzle are high, there is higher possibility of deadlock. The main reason is that more requested items compete for seizing an empty spot. Several situations may result in deadlock as was explained in Chapter 3. However, it was found that the most common deadlock situation is when the distance from the optimal side of two or more requested items to a white cell becomes the same.

The model shows that retrieval time increases when the size of the puzzle linearly. It was also shown that the retrieval time has a logarithmic relationship with the number of cells in a puzzle. This indicates that as the puzzle gets larger, the retrieval time initially increases drastically; however, it levels off at larger values.

The second important finding from the simulation illustrates the impact of empty cells on retrieval time in a specific puzzle configuration. It was found when the puzzle size is smaller (for example 3×3 or 6×6), the curve that presents the average retrieval movements per number of white cells follows 3 distinct trends. The average retrieval movement is almost flat for puzzle configurations with only a few white cells, and then decreases sharply as the number of white cells increases. Toward the tail of the curve, the curve levels again. This means adding more empty (white) cells is not as effective in reducing the average retrieval movements any more. When the puzzle size grows, this pattern repeats itself twice. This means, the average retrieval movement starts out relatively flat for the first few white cells, and then follows a sharp decrease as the number of white cells increases. It levels off again, and then the curve drops sharply for a second time and plateaus at the tail of the curve. This pattern was observable in 20×20 and 25×25 puzzle sizes. Table 4 presents the intervals that increasing the number of white cells appeared insignificant in reducing the average retrieval movements. These intervals are briefly called intervals of indifference. All these findings are simulation results for retrieving 3 items from the puzzle.

Table 4: White cells indifference intervals for simulated puzzle settings

Puzzle size	Intervals of indifference for white cells
6×6	1 to 3, and higher than 25
12×12	1 to 10, and higher than 90
20×20	1 to 3, 10 to 40, and higher than 200
25×25	1 to 3, 10 to 40, and higher than 250

This finding is significant in identifying an optimum number of white cells in a puzzle. Intuitively, it is understandable that maintaining more white cells in the puzzle reduces the retrieval time; however, a high number of white cells reduces storage density. The study shows

that in certain intervals, keeping more empty cells in the puzzle does not support much reduction in retrieval times.

5.2 Recommendations for Future Work

The contributions of this research can be extended by exploring the following options:

1. In conducting this research, the number of requested items were kept constant at 3. This could be extended by measuring the retrieval time when the number of requested items changes as well.
2. This research only investigated item retrievals. An extension to this research can investigate a dynamic network where replenishment occurs concurrent with retrieval.
3. In conducting this research, the system was assumed failure free meaning all cells would make movements required by the system algorithm. An extension to this research could be measuring the impact of grid cell failures on retrieval time.
4. A 3-dimensional storage grid, illustrated in Figure 32, is the ultimate high density storage system that can take advantage of the cubic volume in a warehouse . When space is costly or not available, a 3-D puzzle network that allows the items to move in 3 dimensions and can be retrieved or replenished from all sides of the puzzle is essential.

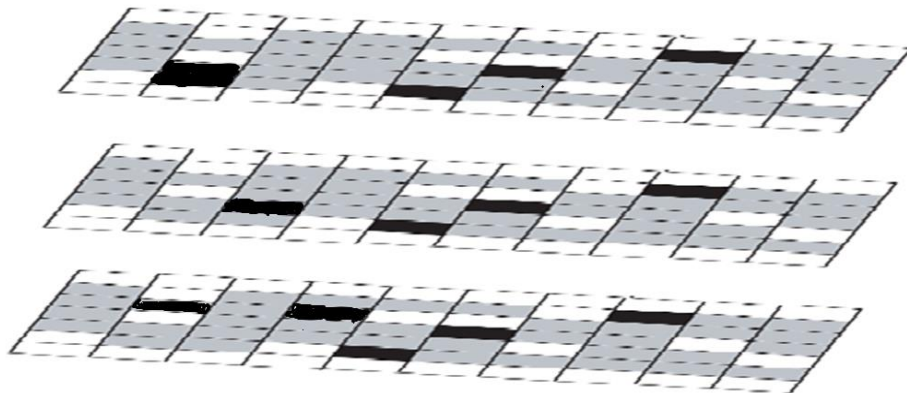


Figure 32: Several levels puzzle system

REFERENCES

- Alfieri, A., Cantamessa, M., Monchiero, A. and Montagna, F., 2012. Heuristics for puzzle-based storage systems driven by a limited set of automated guided vehicles. *Journal of Intelligent Manufacturing*, 23(5), pp.1695-1705.
- Derhami, S., Smith, J.S. and Gue, K.R., 2017. Optimising space utilisation in block stacking warehouses. *International Journal of Production Research*, 55(21), pp.6436-6452.
- Flake, G.W. and Baum, E.B., 2002. Rush Hour is PSPACE-complete, or “Why you should generously tip parking lot attendants”. *Theoretical Computer Science*, 270(1), pp.895-911.
- Furmans, K., Gue, K.R. and Seibold, Z., 2013. Optimization of Failure Behavior of a Decentralized High-Density 2D Storage System. In *Dynamics in Logistics* (pp. 415-425). Springer, Berlin, Heidelberg.
- Furmans, K., Nobbe, C. and Schwab, M., 2011. Future of material handling—modular, flexible and efficient. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Furmans, K., Schönung, F. and Gue, K.R., 2010. Plug-and-work material handling systems. *Progress in material handling research*, pp.132-142.
- Gue K. R., 2006. Very high-density storage systems. *IIE Transactions* 38(1):79–90.
- Gue K. R., Furmans, K., Seibold, Z., Uludağ, O., 2014 GridStore: A Puzzle-Based Storage System With decentralized Control, *Automation Science and Engineering*, *IEEE Transactions on* (Volume:11 , Issue: 2, pages: 429 - 438), ISSN: 1545-5955,
- Gue, K.R. and Furmans, K., 2011, January. Decentralized control in a grid-based storage system. In *IIE Annual Conference. Proceedings* (p. 1). Institute of Industrial and Systems Engineers (IIE).
- Gue, K.R. and Kim, B.S., 2007. Puzzle-based storage systems. *Naval Research Logistics* (NRL), 54(5), pp.556-567.
- Gue, K.R., Uludag, O. and Furmans, K., 2012. A high-density system for carton sequencing. In *6th International Scientific Symposium on Logistics*, Hamburg, Germany.
- Furmans, K., Gue, K.R. and Seibold, Z., 2013. Optimization of Failure Behavior of a Decentralized High-Density 2D Storage System. In *Dynamics in Logistics* (pp. 415-425). Springer, Berlin, Heidelberg.
- Hao, G. and Gue, K.R., 2016. A Two-Sided, High-Density Rail-Rail Hub.

Hearn, R.A. and Demaine, E.D., 2005. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2), pp.72-96.

Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Wooldridge, M.J. and Sierra, C., 2001. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, 10(2), pp.199-215.

Kendall, G., Parkes, A.J. and Spoerer, K., 2008. A Survey of NP-Complete Puzzles. *ICGA Journal*, 31(1), pp.13-34.

Krühn, T., Sohrt, S. and Overmeyer, L., 2016. Mechanical feasibility and decentralized control algorithms of small-scale, multi-directional transport modules. *Logistics Research*, 9(1), p.16.

Material Handling Institute (MHI), 2014, Material handling and logistics, US Roadmap, Version 1.

Mayer, S. and Furmans, K., 2010. Deadlock prevention in a completely decentralized controlled materials flow systems. *Logistics Research*, 2(3-4), pp.147-158.

Mirzaei, M., De Koster, R.B. and Zaerpour, N., 2017. Modelling load retrievals in puzzle-based storage systems. *International Journal of Production Research*, pp.1-13.

Rohit, K.V., Taylor, G.D. and Gue, K.R., 2010, January. Retrieval time performance in puzzle-based storage systems. In *IIE Annual Conference. Proceedings* (p. 1). Institute of Industrial and Systems Engineers (IISE).

Schmidt, T. and F. Schulze (2009). Future approaches to meet complexity requirements in material handling systems. *FME Transactions*. 37(4), pp.159-166 (This reference is in both English and the other language, I used English one)

Seibold, Z., 2011. Optimization of Failure Behavior of a Decentralized High-Density Storage System. Thesis, Auburn University.

Seibold, Z., Stoll, T. and Furmans, K., 2013, April. Layout-optimized controlled conveying modules. In *Systems Conference (SysCon), 2013 IEEE International* (pp. 628-633). IEEE.

Sohrt, S., Seibold, Z., Krühn, T., Prössdorf, L., Overmeyer, L. and Furmans, K., 2014. Buffering algorithms for modular, decentralized controlled material handling systems. *System*, 3(4), p.5.

Taylor G. Don, Kevin R. Gue., 2008, "The Effects of Empty Storage Locations on Puzzle-Based Storage Systems", In *Proceedings of the Industrial Engineering Research Conference*, pages 519–523.

Uludag, O., 2014. GridPick: A high-density puzzle based order-picking system with decentralized control (Doctoral dissertation).

Van Den Berg, J.P. and Gademann, A.J.R.M., 2000. Simulation study of an automated storage/retrieval system. *International Journal of Production Research*, 38(6), pp.1339-1356.