A Puzzle-Based Products Sorting Strategy for Logistics Warehouses

Doctoral Thesis

A Puzzle-Based Products Sorting Strategy for Logistics Warehouses

物流倉庫のためのパズルベースソーティングシステムに関する研究

RAJI ALAHMAD

March, 2022

Department of Life Science and Systems Engineering

Graduate School of Life Science and Systems Engineering

Kyushu Institute of Technology

A Doctoral Thesis

submitted to Graduate School of Life Science and Systems Engineering

Kyushu Institute of Technology

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Engineering

RAJI ALAHMAD

Thesis committee:

Professor Kazuo Ishii	(Supervisor)
Professor Hiroaki Wagatsuma	(Co-Supervisor)
Professor Hiroyuki Miyamoto	(Co-Supervisor)
Professor WATANABE Keisuke	(Co-Supervisor)

ACKNOWLEDGMENTS

In the Name of Allah, the Most Gracious, and the Most Merciful

Foremost, I would like to thank and praise the almighty Allah who gave me so much blessed, strength, and composure all the time especially during my Ph.D.

I would like to express my deep gratitude to my supervisor Prof. Kazuo Ishii for giving me the opportunity to pursue my study in his beloved laboratory. His great support and invaluable assistance were always motivated me to move forward without any worries. His deep knowledge allowed me to gain significant experience through frequent and fruitful discussions. He was a great example of a professional researcher whom I would like to follow in the future.

I would like to thank the committee members Prof. Hiroaki Wagatsuma, Prof. Hiroyuki Miyamoto, and Prof. Keisuke Watanabe for the invaluable comments which enriched my work. I express my thanks to Prof. Yuya Nishida and Prof. Shinsuke Yasukawa for the precious feedback and advice they provided during the meetings.

I would like to express my gratitude to Prof. Doobsub James Jahng who played an important role in my life. The inestimable knowledge he gave me is positively influenced my way of thinking and acting.

A very great thanks to my friend Enrico, he was more than a companion, he was a brother. He was always beside me. Grazie di Cuore.

I would like to thank my dear friend Sung min Nam for the enjoyable conversations on many topics.

A great thanks to all my friends and colleagues in the Ishii laboratory for the great times we spent during trips as well as research.

A great thanks to my dear friends Belal, Obada, Ahmed, Khaled, Farid, and all the Arab community. They were very great supports during my journey in Japan.

I would like to thank the chairman of the Japanese Syria Friendship association (JSFA), Dr. Mohammed Shihab for the priceless support and motivation.

I take the chance to express my thanks to the Rotary Yoneyama Memorial Scholarship for their significant financial support. Great thanks to Mr. Takashi Koga and his wife for the joyful times we spent.

This work was supported partly by the collaborative research project between the Center for Socio-Robotic Synthesis, Kyushu Institute of Technology and Beyond Co., Ltd.

last but not least, I would like to express my deepest gratitude to my family. All my achievements are wholeheartedly dedicated to them. Without their priceless support, I would not be able to stand up. My parents, my brothers, and my sisters.

All praise to Almighty Allah firstly and lastly

ABSTRACT

The new challenging demands of the current market including space should be satisfied by designing modern material flow systems, with higher levels of flexibility and reliability. Designing warehouses using effective material handling equipment such as multi-directional conveyors significantly reduces the cost towards efficient space utilization and time-saving. Several storage strategies can be applied depending on service concerns and products storage conditions, for instance, for storing frozen items that need specific temperature conditions, the zoning strategy is applied. On the other hand, different order picking policies might be used such as Batch picking where the orders would be batched together and the picking process carried out for whole required orders in a single picking round. Under batch and/or zoning picking policy, which is applied in most online retailers' warehouses, products necessitate further processes such as consolidation, sorting, and sequencing. Sequencing of items is one of the important processes that lead to enhancing logistic operations. However, current approaches are not capable of fully fulfilling the dynamic changes, and therefore puzzle-based sequencing system with very high density and highly efficient floor space utilization has been successfully developed.

Accordingly, two puzzle-solving methods are investigated; the game tree and the pathfinding algorithms. A-star is chosen based on pathfinding algorithms in order to find the shortest solution of the puzzle in which the sequencing time is decreased. Furthermore, the pre-sorting process is proposed to overcome the unsolvable configuration issue. The shape of the puzzle is discussed with several factors that affect the sorting steps, and numerically we found that the square shape is better than the rectangular one in terms of

solution steps. Three introduced technical solutions strategies are proposed to increase the limitation of the puzzle; increasing the puzzle size, using multi-boards with the same puzzle boards sizes, and adding buffer conveyor. These strategies are explained and discussed in terms of the area used by the system and the total solution steps. Using multi-boards with the 8-puzzle board size was superior to other strategies. An arbitrary number of blanks in the puzzle was discussed with their effect on the puzzle capacity and maximum solution steps. Moreover, by carrying out double switching in one step with applying the block movement concept, the solution steps are minimized by a minimum of 1 step, an average of 4 steps, and a maximum of 10 steps in an 8-puzzle with 2 blanks placed in the corner of the puzzle, and the average reduction percentage of solution steps was 25%. The best strategy to sequence more than 8 boxes in one sequencing time is using multi-boards along with the main feeding conveyor with the shape and size of 8-puzzle with 2 blanks.

The findings suggest that a puzzle-based sequencing system would be preferred for highly efficient floor space utilization as well as lower sequencing time compared to other systems.

Keywords: Warehouse; Sequencing; 8-puzzle; A-star algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Logistics operations can be elucidated by several fixed assets: warehouses, depots, transport, and material handling. The number and size of these assets are important factors in effective logistics planning [1].

The warehouses take second place in the logistics functions after transport, and its capital and operating cost embody 23% of logistics costs in the US, and 39% in Europe [2].

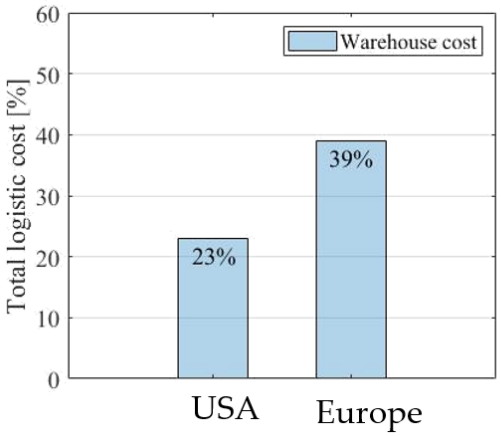


Figure 1.1. The capital and operating cost of warehouses.

Warehouses are often one of the most costly elements of the supply chain [1]. Two types of warehouses can be categorized; distribution warehouses, where the products are collected from the point of origin for delivery to consumers, and production warehouses, where the raw materials and semi-finished

products of production facilities are stored [3]. The proper design of warehouses is one of the most important factors affecting space utilization, efficiency, and cost [4][5]. Figure 1.2 illustrates the common activities of warehouses, which can be summarized in four main parts: receiving, storage, order picking, and dispatching.

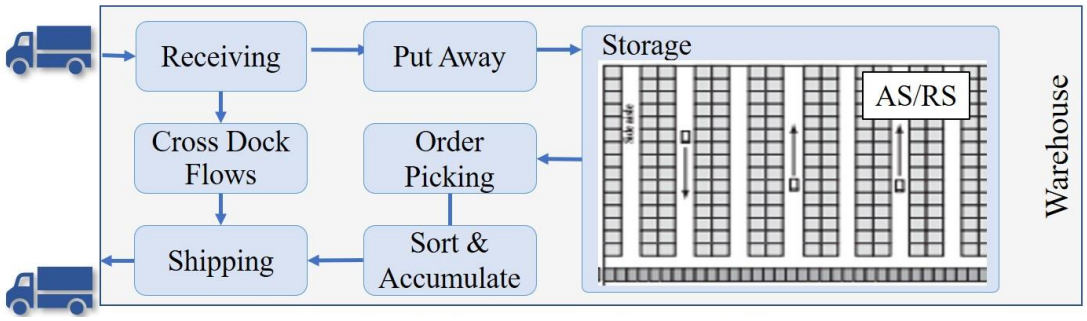


Figure 1.2. The common warehouses' activities.

1. Receiving: This typically includes the physical unloading of incoming transport, and recording the incoming goods into the computer system. As well as the quality control checks which may be undertaken as part of this activity.
2. Storage: Goods are normally taken to the reserve storage area, which is the largest space user in many warehouses. Different storage strategies can be applied depending on service concerns, and goods storage conditions, for instance, for storing some foods and frozen materials which need specific temperature conditions, the zoning strategy is applied. Another strategy might concern customer service, in such a strategy the main concern is to fulfill the delivery time. To do so, the items which are usually ordered by the same customer are stored in the same area in the storage, or store the items that have a higher ordering ranking are stored in the nearest part to the storage output.
3. Order picking: when an order is received from a customer, goods need to be retrieved from the storage area in the correct quantity and

in time to meet the required service level. Depending on the order lists, an order can be retrieved as a full pallet or a sequence of individual items. The warehouse management system gives the order list as well as the location of the items to the picker. Several picking concepts can be applied in the warehouses, for example:

- Pick-to-order: basically, when the picker takes one order and travels through the warehouse until picking all order items. The main disadvantage of a pick-to-order policy is that pickers have to travel for every single order, this policy would be very inefficient, especially in situations where the range of the products is very large.
- Batch picking: in this regime, the orders would be batched together and the picking would be conducted for whole required orders in a single picking round. This is very common, particularly for small orders.

4. Dispatch: Goods that accumulated together are loaded onto outbound vehicles for onward dispatch to the next 'node' in the supply chain, for example to another distribution center or customer delivery vehicles.

The effective use of space is a goal for almost every company located near population centers, where high space charges and limited availability of real estate are the main concern [6]. Smaller warehouse systems decrease the overall costs since they are less expensive to build [7].

Material handling is the movement of raw materials and semi-finished and finished products to and from productive processes, in warehouses and receiving and dispatching zones [3], and its activities consume 20% to 50% of the total operating costs. Effective material transport equipment, such as rollers, wheels, and sorting conveyors, lead to significant cost reductions and efficient space utilization [8, 9].

For efficient warehousing (i.e. put-away, storage, and order picking), an Automatic Store and Retrieval System (ASRS) is typically used [10].

AS/RS is operated by computer control, the controlled cranes run up and down to put away and extract pallets which are in face occupy about half of the stored goods [1]. These cranes are electrically powered and run on rails, positioned on the floor, and are guided by a further rail above the top rack.

In an ASRS, cranes operate in parallel and feed the pallet building workstation; therefore, the robotic palletizer receives a random sequence of items that should be re-sequenced [11].

In the warehouses where zoning strategy is applied, the orders are picked in different zones at the same time, therefore, the outcoming items may need to be consolidated. In addition, applying the batch picking policy leads to the necessity of unpacking, sorting, and resequencing the items of each batch. Referring to the systems that are applied in real-world warehouses, the items are mostly released from ASRS in random sequence [12, 13]. Thus they need either optimized release (which is still under research and development [13]) or items re-sequencing after retrieval for better performance. Especially during peak hours, where a lack of workforce and other new technologies are highly required at the packing stations to timely release the lanes.

Furthermore, mixed-model assembly lines (figure 1.3) have become common in the automotive industry, and the efficiency of the final assembly depends on the sequence of vehicles being built [10].

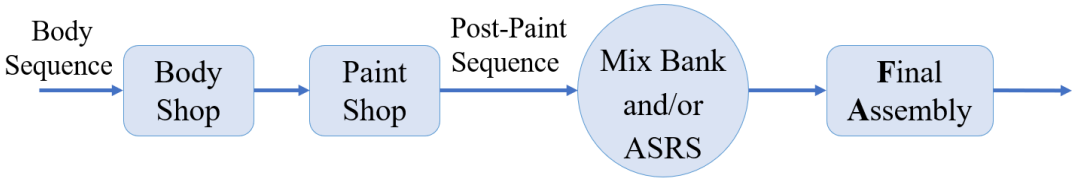


Figure 1.3. Mixed model assembly line.

1.2 Problem Statement

Several systems are used to re-sequence the outcoming random items, for instance:

a) A temporary storage system that uses parallel lanes called mix bank [10]. Here, items inter the system in random sequence, and they are sorted in different lanes to be retrieved in the desired sequence as figure 1.4 illustrates.

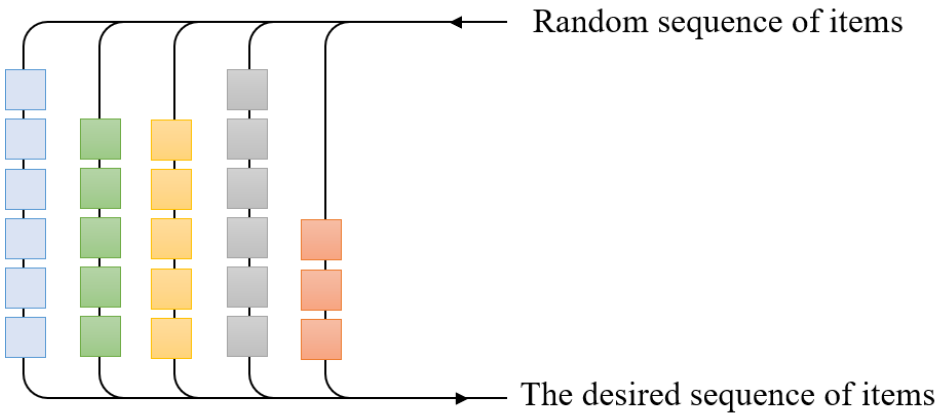


Figure 1.4. 5 lanes mixed bank.

b) A sortation conveyor, where the items keep looping until they are in the desired sequence [14]. Usually, this system is used for sorting items into different gateways for different output destinations, however, this system is also sued for re-sequence random items. Figure 1.5 shows the sortation conveyor system.

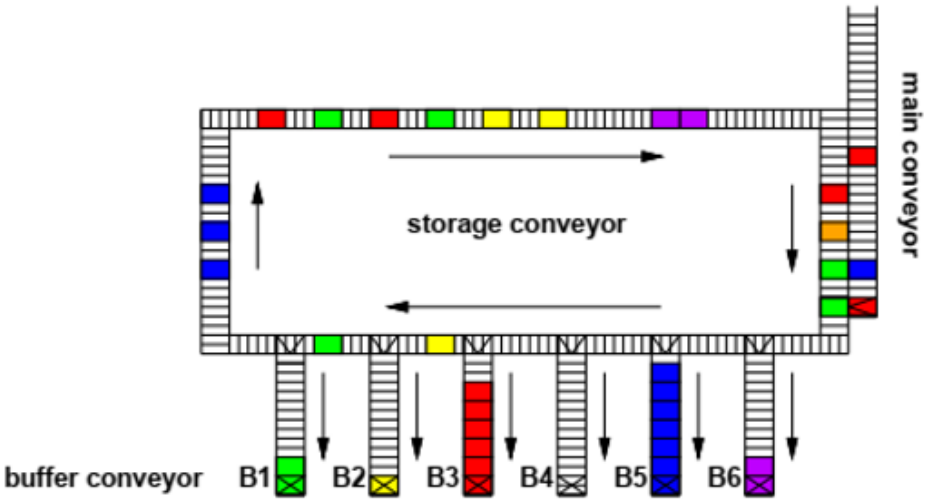


Figure 1.5. Sortation conveyor [14].

Since such systems need a large area, the poor floor space utilization is one of the disadvantages of these systems [11]; therefore, a material handling device with a high-density system is required as well as the sequencing time should be minimized to provide higher throughput, these two main points are important concerns for designing the warehouses.

Generally, the term density in logistics is used for storage density which is the ratio of storage area to the total warehouse space[15]. However, in this study, the density is defined as the areal density which is the ratio of items to the total material handling device space.

1.3 Literature Review

In this section, we will investigate the literature from two different points of view: considering the material handling technologies that are applied in the warehouse which could carry out the sequencing process. And the high-density systems in warehouses which can provide a high space utilization.

1.3.1 Material Handling Technologies

The material handling activities consume 20% to 50% of the total operating costs as mentioned in Chapter 1. Effective material handling equipment plays a key issue in enhancing the warehousing activities. In the warehouses, there are two main material handling technologies that can carry out the sequencing process: conveyors and small-scaled multi-directional conveyor systems.

- **Conveyors**

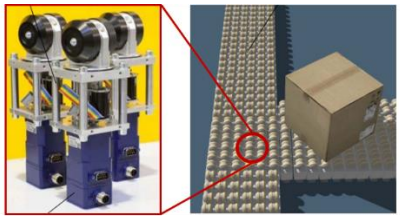
  the conveyor system considers as the most common material handling equipment in the warehouses. Both gravity and powered conveyors can be used for moving the goods between two fixed points. Typically, the gravity conveyor systems include chutes, skate-wheel conveyors used to move the goods for short distances, and the powered conveyor systems include Roller and belt conveyors used for long distances.

  In principle, the conveyor system is characterized in a way to fulfill simple intralogistics tasks, for instance, moving the goods on a straight line. However, for more complex tasks such as rotation and sorting, the conveyor system must be extended with additional mechanical components or modules [16]. This makes the conveyor technology rigid, less maintenance-friendly, and cost-intensive. For these reasons, we sought the possibility of redesigning the conveyor system.

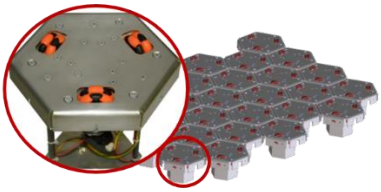
- **Small-Scaled Multi-directional Conveyor Systems**

  To fulfill the demands of intralogistics in terms of material flow, small-scale modules might be applied where the conveyed products are bigger than one module in the system.

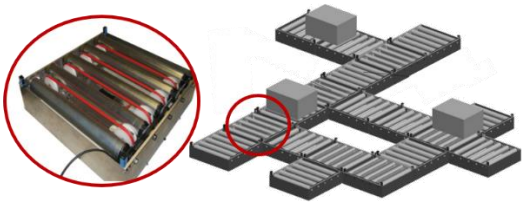
  Figure 1.6 illustrates CogniLog, Flexconveyor, and Celluveyor modules which are some small-scale systems.



(a) CogniLog system



(b) Celluveyor modules



(c) Flexconveyor

Figure 1.6. (a) CogniLog system [17], (b) Celluveyor modules [18], (c) Flexconveyor System [19] .

As shown in the figure, Celluveyor is a unique modular made of several small hexagonal modules, each consisting of three omnidirectional wheels independently driven. Multi-Functionalities can be controlled only via software without the need for mechanical modifications.

Based on such high flexible technologies, many systems have been developed for high density and space utilization.

1.3.2 High-density Systems

Many studies have considered high-density systems in order to enhance the efficiency of logistics processes. The sliding puzzle was invented by Sam Loyd in the 1870s [7], and is also known as the 15-puzzle, and later, the general version ($n^2 - 1$) became a popular and interesting subject for logistics researchers, especially in developing storage systems.

In fact, the puzzle concept was the inspirit of many researchers to invent and develop systems with high-density to enhance the warehousing functions.

Gue [6] developed a new concept based on a puzzle game: a very high-density storage system (HDSS) for physical goods with an efficient algorithm for filling densely rectangular storage areas. Later, Gue and Kim [7] developed an algorithm for the retrieval of items in a puzzle-based storage system (PBSS). They experimentally compared puzzle-based with traditional aisle-based storage. The results showed that the puzzle-based system was superior, with multiple escorts regarding the retrieval time, if the storage density was less than 90%. In [20], Kota et al. extended the analytical results of retrieval time in PBSS to determine the retrieval time performance when multiple escorts are randomly located within the system. The GridStore system was developed by Gue et al. [21] to overcome the inflexibility of automated material handling systems for HDSS by implementing decentralized control. In GridStore, an arbitrary number of requests could be retrieved by allowing simultaneous item moving. The major drawback of this system is the capability of delivering items to only a single side. However, Uludag [22] solved this limitation by developing a puzzle-based order picking system called GridPick. In the GridPick system, the orders can be picked from two sides of the grid, allowing for higher throughput and efficient use of space compared to single-sided systems. Figure 1.7 shows GridStore and GridPick systems.

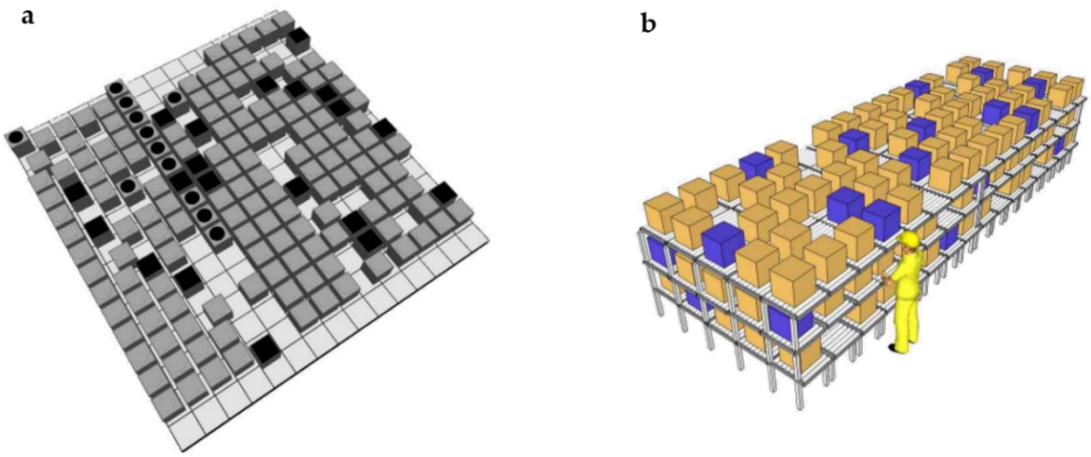


Figure 1.7. a) GridStore system, b) GridPick system [23].

A further improvement was achieved by Gue and Hao [24]. They developed a new system called GridHub, which was able to transfer orders in four directions simultaneously within the grid. Subsequently, Hao [23] developed the NU GridHub system to handle bigger boxes in which one box can occupy more than one conveyor module. Further modification of GridHub was conducted by Ashgzari and Gue [25]. Figure 1.8 shows the GridHub and NU GridHub systems.

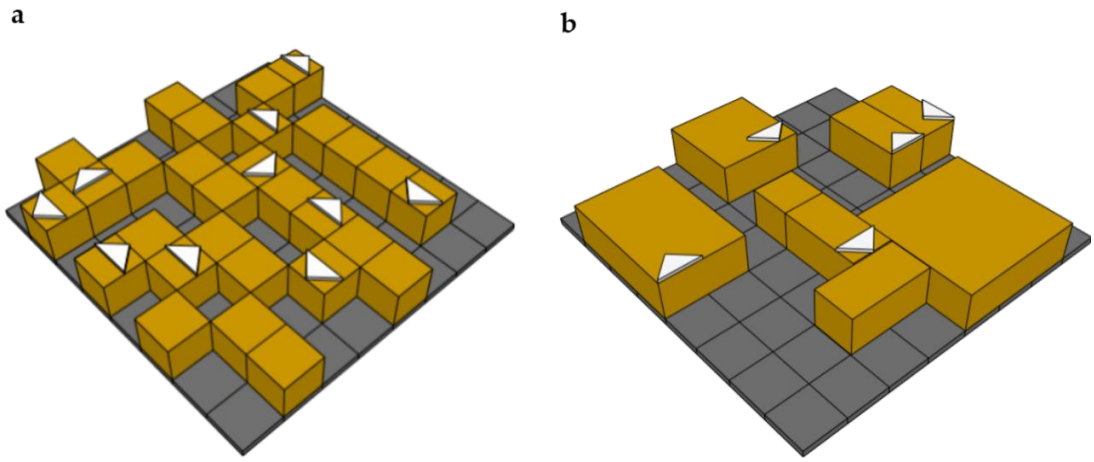


Figure 1.8. GridHub system, b) NU GridHub system [23]

In the new method, GridPick+, several limitations of GridPik were addressed. For instance, GridPick+ allowed the requested items to be delivered into

specific picking positions on the edge of the grid. Moreover, the use of the sequencing function allowed multiple orders to be processed simultaneously. An algorithm for moving several items at the same time in grid-based storage was designed by Yalcin et al. [15] by avoiding the items' conflict. Their experimental results demonstrated that for storage, the pushback strategy achieved the shortest time and distance, and the puzzle-based retrieval strategy was most efficient. Yalcin et al. [26] also addressed the problem of item retrieval from puzzle-based storage with a minimum number of item moves. In this work, they proposed an exact search algorithm with several search-guiding estimate functions. Additionally, they discussed the configurations with multiple empty cells located in the grid with different grid sizes.

In recent research, Shirazi and Zolghadr [27] developed an algorithm for item retrieval for HDSS. This method guaranteed the deadlock freeness in the algorithm and discussed different puzzle sizes with a dissimilar number of empty cells. It was observed that increasing empty cells up to three cells will increase the average retrieval movement, while increasing the empty cells above three will decrease the average retrieval movement sharply. Further research was carried out to formalize arranging smart boxes into an autonomous delivery vehicle [28]. The authors proposed the snake-line concept utilizing the puzzle arrangement to find the tradeoff between space and access rapidity and were able to guarantee the boxes moving continuously with minimum movement.

The system we proposed in this paper was compared with the high-density systems described in the literature, as illustrated in Table 1.1. The used system, function, contribution, and system areal-density are listed in the table to distinguish these works.

Table 1.1. Comparison among the proposed system and other high-density systems from the literature.

System	System	Function	Contribution	System Areal-Density for 35 Boxes
Gue and Kim [7]	NAVSTORS system	Storing, retrieval	Describe the relationship between storage density and expected retrieval time	94.4% with two escort
Gue et al. [11]	GridSequence	Sequencing	High density, a decentralized control algorithm	72.9%
Kota et al. [20]	Puzzle-Based system	Storing, retrieval	Determine the retrieval time performance for multi-escorts randomly located in the grid.	94.4% with two escorts [1]
Gue et al. [21]	GridStore system	Storing, retrieval	Retrieve several items by allowing simultaneous moving	≤94.4% [1]
Uludag [22]	GridPick	Storing, retrieval	Higher throughput, retrieve items to two sides of the grid	≤94.4% [1]
Gue and Hao [24]	GridHub	Storing, retrieval	Transfer orders in four directions simultaneously within a grid	≤95.45% [2] ≤94.44 for 36 boxes
Hao [23]	NU GridHub	Sorting, sequencing	Delivers requested items in the desired sequence to any location	56.25% for 36 boxes
Ashgzari et al. [25]	GridPick+	Storing, retrieval	Increasing in throughput by 77%	-
Yalcin et al. [15]	Grid-based system	Storing, retrieval	Framework for the efficient storage and retrieval of items based	Up to 100%

			on a multi-agent routing algorithm	
Yalcin et al. [26]	PBS system	Items retrieval	Retrieve items with a minimum number of items moves	≤94.4% [1]
Shirazi et al. [27]	PBS system	Items retrieval	Deadlock prevention algorithm	Up to 97.2%
Tetouani et al. [28]	Puzzle-based system	Rearrangement while Routing" strategy	Formalize arranging smart boxes in an autonomous delivery vehicle	97.2%
Proposed method	Puzzle-based sequencing system	Sequencing	High-density sequencing system, address unsolvable puzzle configuration	97.2%

[1] Since these systems involve the puzzle-based concept, the areal-density is calculated as $(n_c - \mathrm{e})/\, n_c$, where n_c is the number of grid cells and e is the number of empty spaces in the grid.

[2] One rule of GridHub is that at least one empty module has to be in each column or row, and their experiment was set as a grid with 22 columns and 11 rows.

Although several studies have considered high-density and puzzle-based systems with their applications, most of them have focused on storage and item retrieval. In these systems, the items are retrieved in the desired sequence. However, under batch and/or zoning picking policy, which is applied in most online retailers' warehouses, items necessitate further processes such as consolidation and sequencing [13]. To the authors' best knowledge, very few contributions have been published in the literatures that have addressed the issue of item sequencing, for instance, GridSequence, which was developed by Gue et al. [11]. The proposed system could re-sequence incoming items to feed a palletizing robot with the required sequence. The GridSequence system consists of a puzzle grid with (n × m) dimensions, plus one additional row and

one additional column; thus, the whole system dimensions are (n + 1) × (m + 1) as illustrated in figure 1.9.

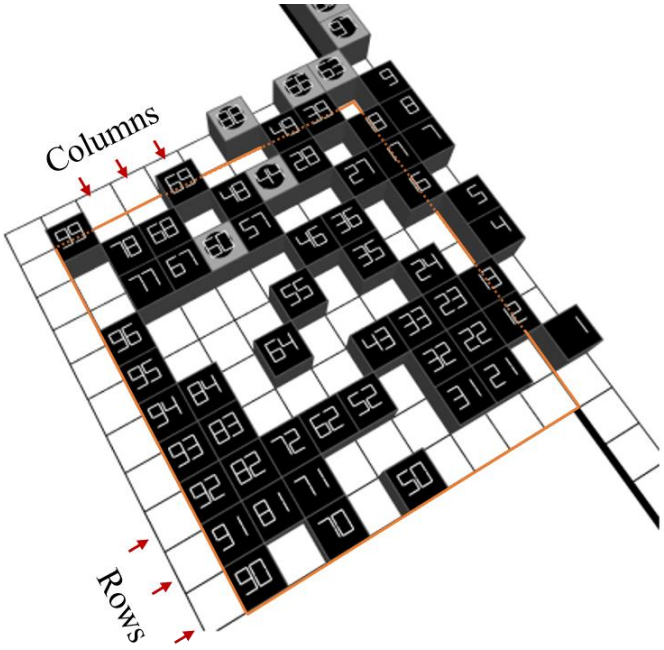


Figure 1.9. GridSequence system [11]

The authors showed the effect of the aspect ratio on the sequencing time in the experimental results and suggested that the aspect ratio should be at least 10. Furthermore, adding one more additional column to the center of the grid can positively affect the system. The major drawback of this system is low space utilization, since adding rows and columns will occupy more spaces out of the grid, and decrease the density. A lower density means higher empty spaces in the grid and an increase in floor space usage. Thus, the density plays a key role in evaluating the utilization of floor space of warehouses (storage and other functions) in urban areas where the limited space should be utilized efficiently.

1.4 Research Objectives

We can summarize the objectives of this thesis in the following points:

1. To realize a high-density sequencing system based on the puzzle movement concept, with highly efficient floor space utilization concerning the minimum item movements. These points are directly related to better energy efficiency and, consequently, to lower operational costs. The analysis here carried out represents a tool for improving the warehouse activities in terms of both space utilization and time consumption, in addition to minimizing the workforce.
2. To propose the puzzle-solving algorithm to fulfill the sequencing process.
3. To set up an optimal design of sequencing board in shape, size and the number for the practical implementation of a real-world warehouse.

1.5 Concept of Puzzle-based Sequencing System

Using ASRS in the storage can increase the efficiency of warehousing functions because this system approaches seven-day-week, 24-hour operations. The cranes of this system work in parallel in both in-feed and out-feed. therefore, the outcoming boxes come in a random sequence. These boxes are moved on the conveyor and inter the proposed sequencing system which is the puzzle, afterward, the puzzle starts the sequencing process to reach the goal configuration. Finally, the boxes come out as a series of boxes with the desired sequence. Figure 1.10 illustrates the proposed sequencing system concept.

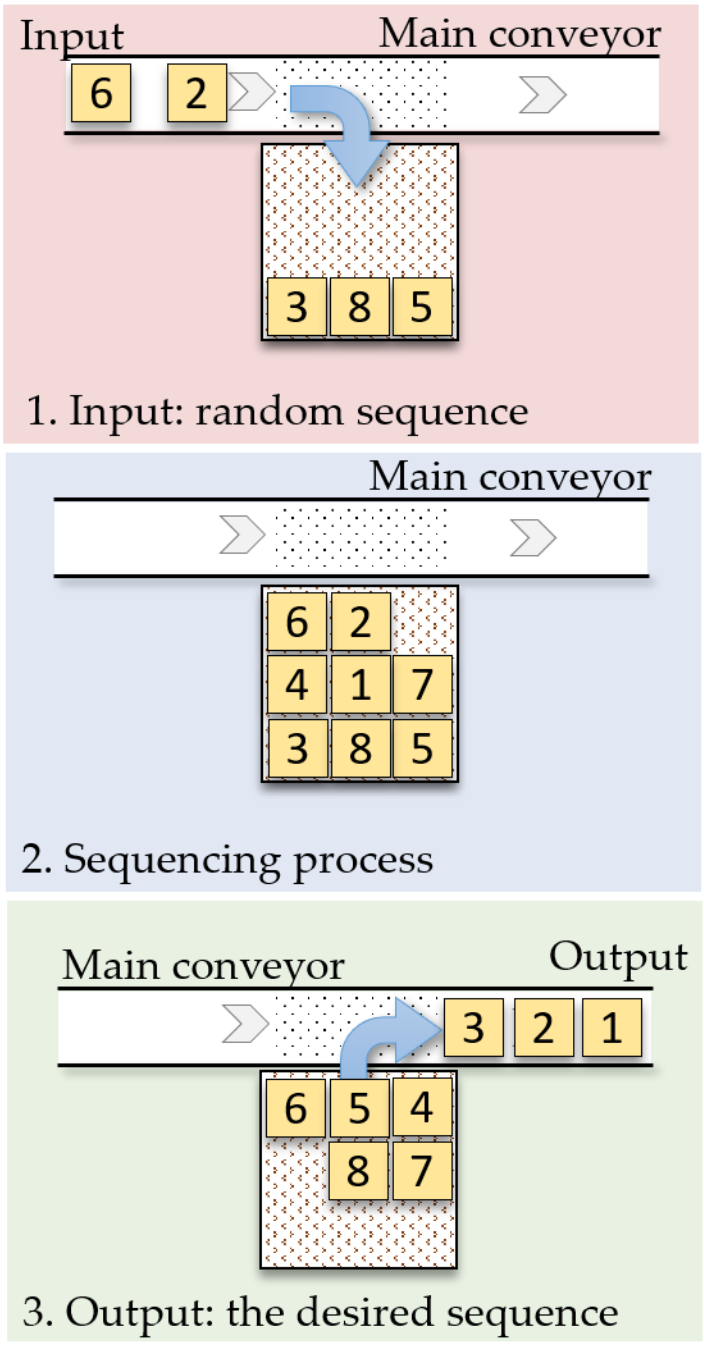


Figure 1.10. The proposed sequencing system concept.

According to the figure, the series of boxes come from the storage out of sequence, and inter the puzzle board. Then, the sequencing process starts until we get the goal configuration. Finally, the boxes outcome from the board inter the main conveyor as a series of sequenced boxes and move into the next process in the warehouse.

the following assumptions are made for this system:

1. One set of the sequencing system is 2 dimensions, so the process doesn't consider the 3D cubic accumulation problem.
2. The system is used under the zoning storing strategy where the boxes come separately, and/or under batching picking policy where the order comes as a batch and the boxes of each batch are separated into a series of boxes.
3. All boxes are square-shaped and have the same base area.
4. The sequencing process in the puzzle starts after filling in the puzzle board with all boxes.
5. Incoming boxes to the board inter one by one, while in the output, the boxes are out as row by row as shown in figure 1.10. (3).
6. We allow simultaneous moving so the boxes are moved into the board simultaneously. the same during the output process.

1.6 Layout of the Thesis

The thesis includes four chapters that are structured as follows:

Chapter 1. background, the literature review, and the research objectives are presented.

Chapter 2. this chapter presents the methodology of the research starting with an investigation of the puzzle-solving methods. Two solving methods were investigated: game tree and pathfinding algorithms. A-star was chosen based on pathfinding algorithms in order to find the shortest solution of the puzzle in which the sequencing time. In this chapter A-star algorithm was explained in detail with a proposal of a pre-sorting strategy to overcome the unsolvable configuration issue that cannot be solved by the aforementioned methods.

Different shapes of the puzzle can carry out the sequencing process, thus, two different shapes, in particular, square and rectangular shapes, were discussed. In addition, the factors that affect the number of solution steps. Furthermore, three proposed strategies to fulfill the practical implementation in the warehouse are presented in this chapter. In this Chapter also, the effect of increasing the number of blanks in the puzzle on the system is presented. Furthermore, more blanks in the puzzle allowed a double-switching process which reduced the maximum number of the solution steps.

Chapter 3. the results and discussion of the points presented in the methodology are presented in this Chapter.

Chapter 4. finally, the conclusions of the thesis are summarized, and the possible future work is discussed.

1.7 Summary

A comprehensive introduction to develop a high-density sequencing system concerning the minimum sequence g time is outlined in this chapter. The previous research in the field of logistics that considered the high-density system was investigated with a comparison between our proposed method with the previous works in terms of the density and floor space. This study presents a high-density puzzle-based system for products sequencing considering the sequencing time. To complete the proposed method, a large number of investigations with lots of analysis are implemented to provide the sequencing time consuming and compare it with conventional sequencing systems and algorithms.

CHAPTER 2

RESEARCH METHODOLOGY

This chapter presents the puzzle-based system with investigating puzzle-solving methods to choose the best sorting algorithm. Afterward, we set up the sequencing system design with the optimum board shape, size, and number. The optimum parameter of the design was evaluated based on the time which is the movement steps of the boxes on the sequencing board, and the floor area occupied by the sequencing system.

2.1 Sliding Puzzle

As mentioned in Chapter 1, The sliding puzzle was invented by Sam Loyd in the 1870s [7], and is also known as the 15-puzzle, and later, the general version ($n^2 - 1$) became a popular and interesting subject for many researchers.
Generally, the sliding puzzle is a single-agent sliding game consisting of (n × m) − 1 square tile and one blank, distributed in an (n × m) grid. The process for solving this is to rearrange a random configuration of numbers in the initial state by sliding the blank tile in one of four allowable moves (Up, Down, Right, and Left) to reach the goal state, which is the proper sequence of numbers [29], as shown in Figure 2.1.

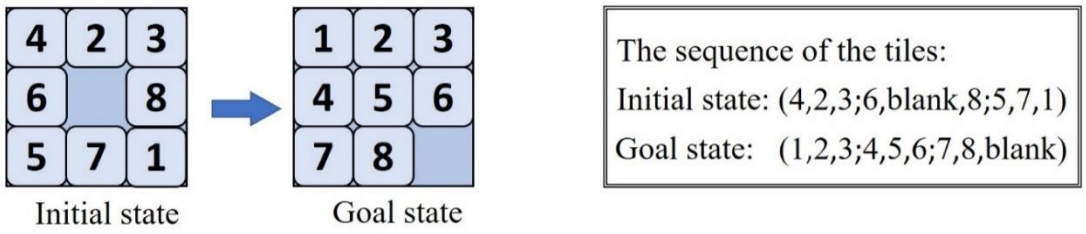


Figure 2.1. 3x3 puzzle (8-puzzle), random configuration (Left), goal state (Right).

There are different shapes and sizes of such a puzzle. The ($n^2 - 1$) puzzle is a specific type, where the board is square (n × n) with ($n^2 - 1$) numbered tiles and one blank [30].

8-puzzle is one of the most famous (n^2 -1) puzzles. Since 15-puzzle and 24-puzzle are extension versions of 8-puzzle. Our study was conducted utilizing an 8-puzzle to simplify the analyses.

2.2 Sequencing Algorithm

There are 9! different configurations of this puzzle, and every second permutation are solvable, Hence, there is a total of 9!=2 =362,880 solvable configurations [31]. Many researchers have an interest in solving such puzzles with the fewest moves (the shortest path to the solution) and they consider finding the optimal solution in two levels, the space and time consuming by the used algorithm, and the number of moves. In this research, we take into consideration the number of moves to reach the goal configuration.

There are two typical methods for finding the shortest path to the solution which achieve the minimum number of tiles moves, game tree, and pathfinding algorithms. In this section, we will discuss both methods in terms of using the puzzle for items sequencing.

2.2.1 Game Tree

This method creates a tree of all configurations (states) that can be generated for the puzzle and finds the target configuration in this tree. In the game tree, all states are represented by nodes, and the depth of the tree denotes the number of solution steps. The procedure is as follows:

1. Start tree creation from the target state configuration;
2. Find the input node (the initial configuration) in this tree; and
3. Trackback the path which leads to the initial node.

The game tree method could guarantee to find the shortest path to the solution. However, we might face two problems: the huge number of states that could be generated, and the scenario of searching for different targets (specific configurations).

I. The Huge Number of States

We start generating the tree by switching the blank with the neighbor tiles. All available switches of one configuration are carried out in one level of the tree (tree depth). Equation (2.1) provides the total number of nodes that could be generated in the tree for the 8-puzzle:

$$\mathbf{N_{States}} = \mathbf{1} + \sum_{\mathbf{i=1}}^{\mathbf{d}} \mathbf{b^i} \ , \qquad (2.1)$$

where N_{States} is the total number of states in the tree; b is the branching factor; and d is the depth of the tree. The branching factor is the number of nodes that could be expanded from the previous node in the tree. For example, if the blank is placed in the corner, the branching factor is 2 since we can switch two tiles, and we get two different states out of the current one as shown in Figure 2.2 which shows the concept of branching numbers for 8-puzzle.

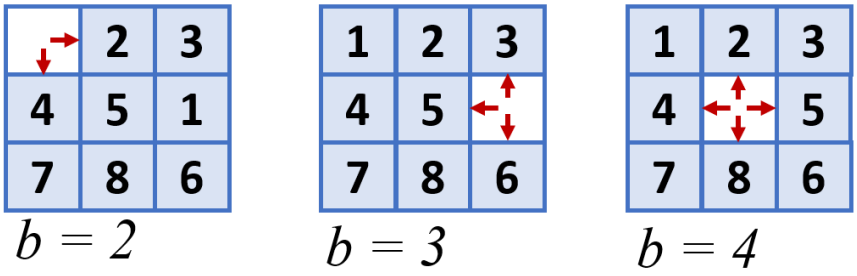


Figure 2.2. 8-puzzle branching factor b.

According to figure 2.2, In 8-puzzle we have three different branching factors. On the top of the tree, we start with a branching factor of 2, since the blank position is in the corner in the goal configuration. Therefore, the first level in the tree has two states, each of which has a branching factor of 3 yielding 6 states in the second level of the tree, for a total of 9 states.

From Figure 2, the branching factor was about 3 (when the blank tile is in the corner, there are two possible moves; when it is along edges, there are three; and when it is in the middle, there are four).

Regarding the depth, Figure 2.3 illustrates the histogram of the solution steps for all solvable configurations of the 8-puzzle as well as the Probability Density Function (PDF) for a normal distribution. We obtained an average solution depth of 22. The same result was confirmed with the work by Reinefeld [32].

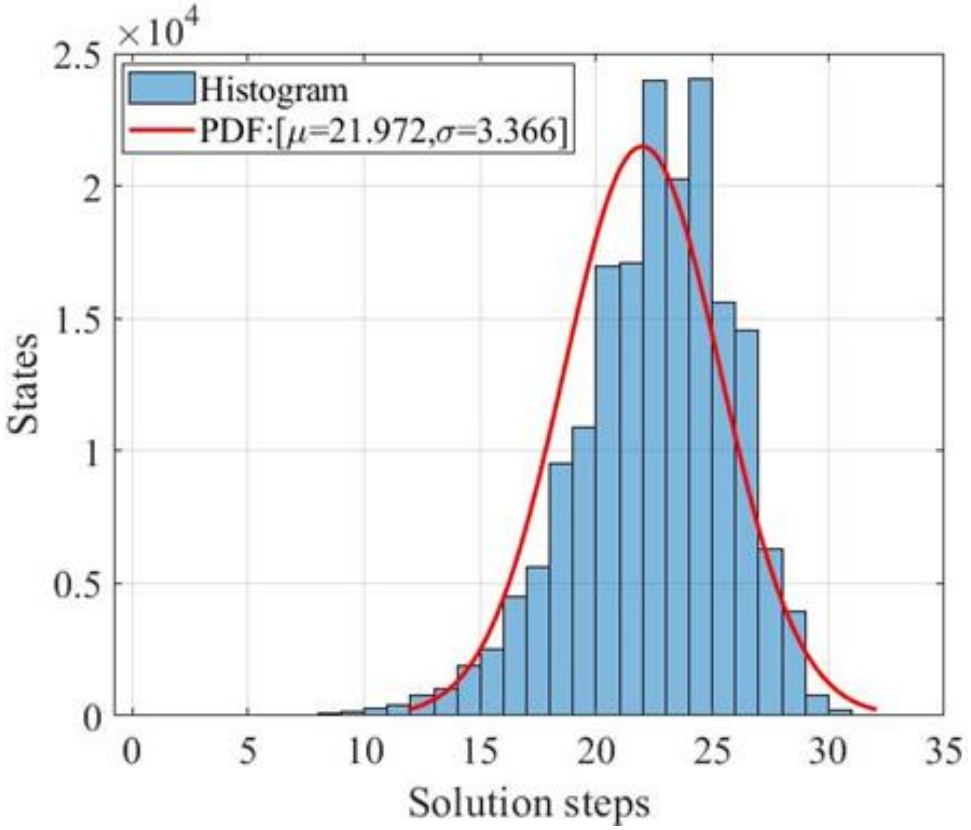


Figure 2.3. Histogram of solution steps for 8-puzzle.

Referring to Equation (2.1), the number of nodes that could be generated for depth 22 and branching factor 3 is 3.13×10^{10} nodes. This huge number of nodes not only requires time to be generated but is also inefficient in terms of memory [33]. By tracking the repeated states, we cut the tree down drastically into 9! / 2 = 181,440 nodes.

II. Searching for Different Targets

In the case of different targets, where the goal configuration is not (1 ,2, 3; 4 ,5, 6; 7, 8, blank), but can be any configuration of 9! States, we need to generate a tree of nodes for each goal. Thus, we had to generate 9! = 362,880 trees and about 13.16×10^{10} nodes in total.

One proposal to overcome the problem associated with generating such a huge number is to search for the input state in the current tree. The following steps describe the concept of searching for a different target in the current tree:

1. Change the desired target to the target in the current tree;
2. Apply the same changes to the input; and
3. Find the new input in the current tree.

Figure 2.4 shows the proposal of searching in the current tree.

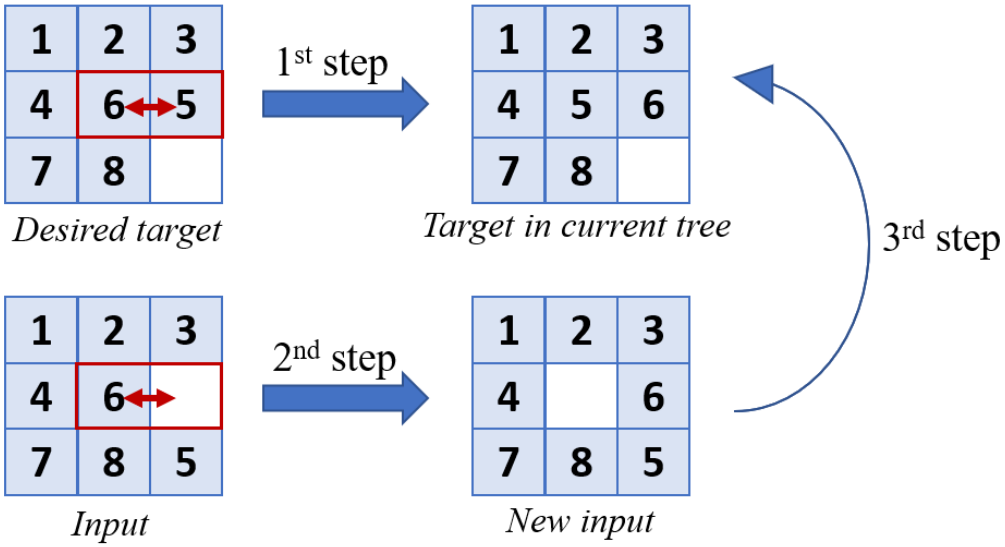


Figure 2.4. Concept of searching in the current tree.

In the example shown in figure 2.4, we first, switch the tiles 5 (numbered 6) and tile 6 (numbered 5) to get the target configuration in the current tree (1, 2, 3; 4, 5, 6; 7, 8, blank). Then we apply the same changes for the input state (1, 2, 3; 4, 6, blank; 7, 8, 5) by switching tile 5 (numbered 6) and tile 6 which is blank. we get the new input (which we are searching for) is (1, 2, 3; 4, blank, 6; 7, 8, 5). Finally, we search for the new input in the current tree.

In this example, the new input configuration is unsolvable, therefore, we cannot find it in the current tree.

Since tiles changing might give unsolvable configurations, this method will not work for all the cases in our system. The solvability of the puzzle is an important concept; therefore, the solvability condition will be discussed in Section 2.5.

2.2.2 Pathfinding Algorithms

To reach the puzzle solution, pathfinding algorithms can be applied by creating a tree of puzzle configurations (nodes), starting from the initial state until the goal state is matched, and then tracking back to the path, which leads to the goal. When reaching the goal state (node), the process of node creation will stop; therefore, generating a huge number of nodes can be avoided. There are two different types of pathfinding algorithms:

I. Uninformed Algorithms (Blind Algorithms)

Such algorithms work without using any external information to guide the agent to reach the goal state. Following are some of such algorithms[33, 34]:

- Breadth-First Search (BFS);
- Depth-First Search (DFS);
- Iterative Deepening Depth-First (IDS).

II. Informed Algorithms

In these algorithms, some information can be used to lead the algorithm and direct it to achieve better performance. This information could be the status and values of the neighbors.

Following are the most common pathfinding algorithms[33, 34]:

- Greedy algorithm;
- A-star (A*) algorithm;
- Iterative Deeping A-star (IDA*) algorithm.

Among the algorithms that extend search paths from the root, A-star is optimally efficient [34, 35]. Hence, A-star was the core algorithm in this study.

2.3 A-star Algorithm

In the A-star algorithm (A*), the nodes can be evaluated using the cost function (Equation (2.2)), which is the sum of two factors: the heuristic function, which estimates how close the current node is to the goal, and the cost from the initial node to the current one [36].

$$\mathbf{f(n) = g(n) + h(n),} \tag{2.2}$$

where f(n) is the evaluation function for the A* algorithm; g(n) is the cost from the initial node to the current node n; and h(n) is the estimated cost from the node n to the target.

Many estimation functions can be used with the A-star algorithm such as Hamming distance and Manhattan distance.

2.3.1 Hamming Distance

This is the count of the number of tiles in the current configuration which are not at the same position as in the goal configuration[33]. Figure 2.5 shows an example of hamming distance.

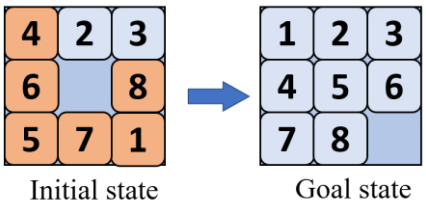


Figure 2.5. An example of hamming distance calculation.

In the example shown in figure 2.5, we note that (4,6,8,5,7,1) tiles are not at the same position as in the goal state. Hamming distance is 6 in this example.

2.3.2 Manhattan Distance

Manhattan distance or city block distance is the absolute vertical and horizontal distance between the tile in the current configuration and its appearance in the goal configuration [33].

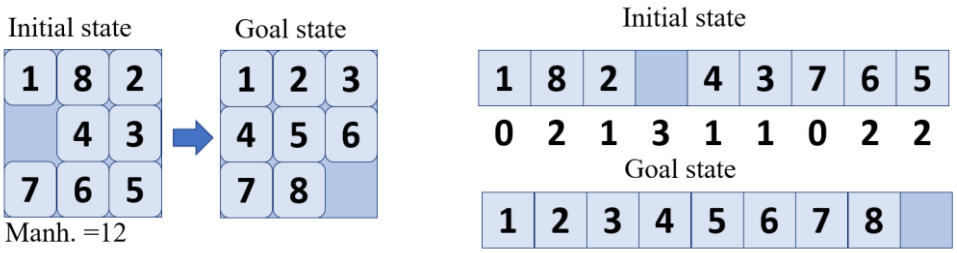


Figure 2.6. An example of Manhattan distance calculation.

The estimation function used in this research was the Manhattan distance, since it showed better performance for the informed search techniques [33, 35]. The Manhattan distance or city block distance is the absolute vertical and horizontal distance between the tile in the current configuration and its appearance in the goal configuration. Figure 2.7 shows the layout of the A-star algorithm for solving the n-puzzle with the fewest solution steps.

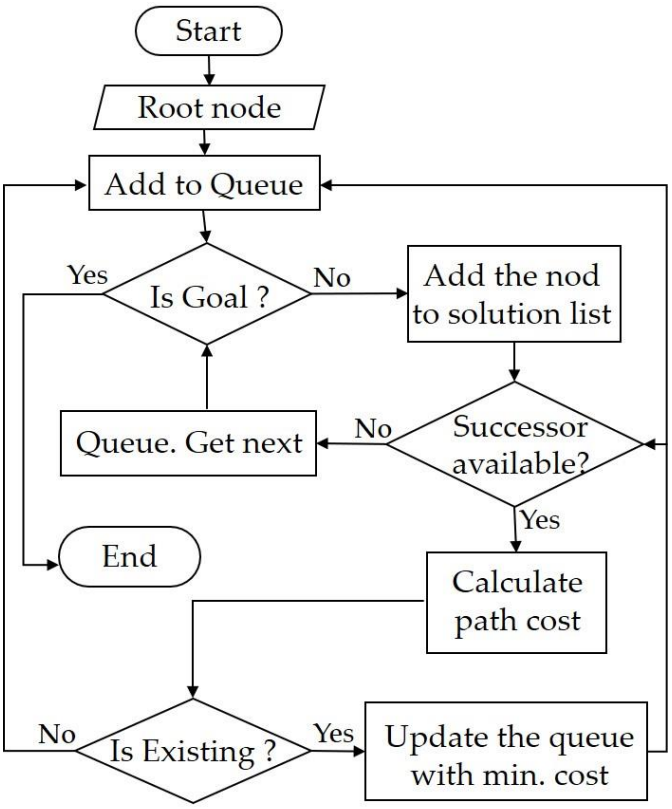


Figure 2.7. A-star algorithm for n-puzzle.

The A-star algorithm allows us to avoid many nodes that should not be selected, avoiding the waste of time caused by searching a large number of useless nodes. The whole search process has strong directionality [37].

Even though the A-star algorithm is optimal for solving the n-puzzle, it was not sufficient for our application, thus we needed to modify it to fulfill the sequencing process.

The reason for the insufficiency of the basic A-star algorithm is the solvability problem. All researchers who are interested in puzzle-solving algorithms have investigated only the solvable configuration of the puzzle, However, in our application, we have a 50% possibility of unsolvable initial configurations of the boxes.

The first modification in the algorithm is checking the solvability condition. as shown in figure 2.8

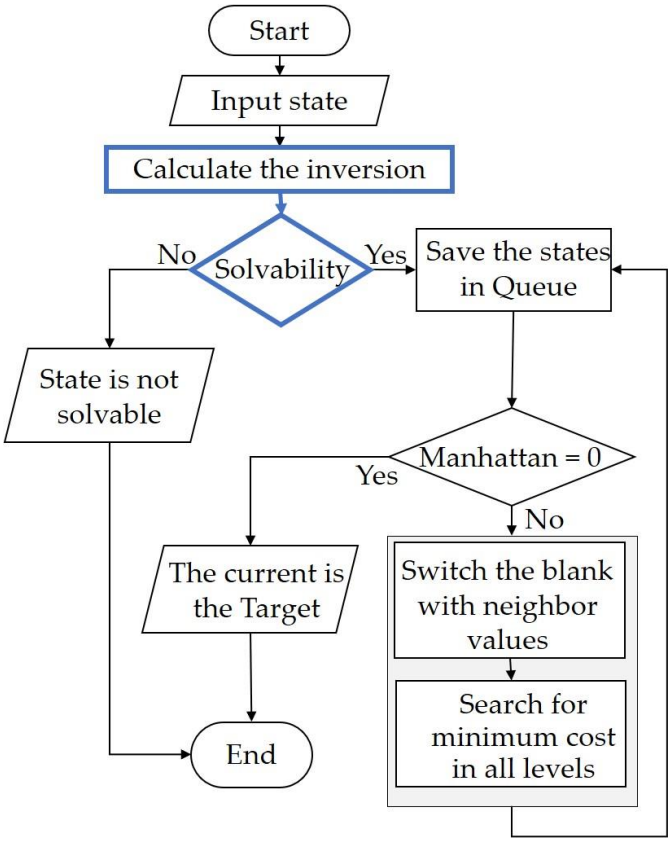


Figure 2.8 A-star algorithm for n-puzzle with solvability condition.

Figure 2.9 illustrates the implementation of This A-star algorithm for the 8-puzzle.

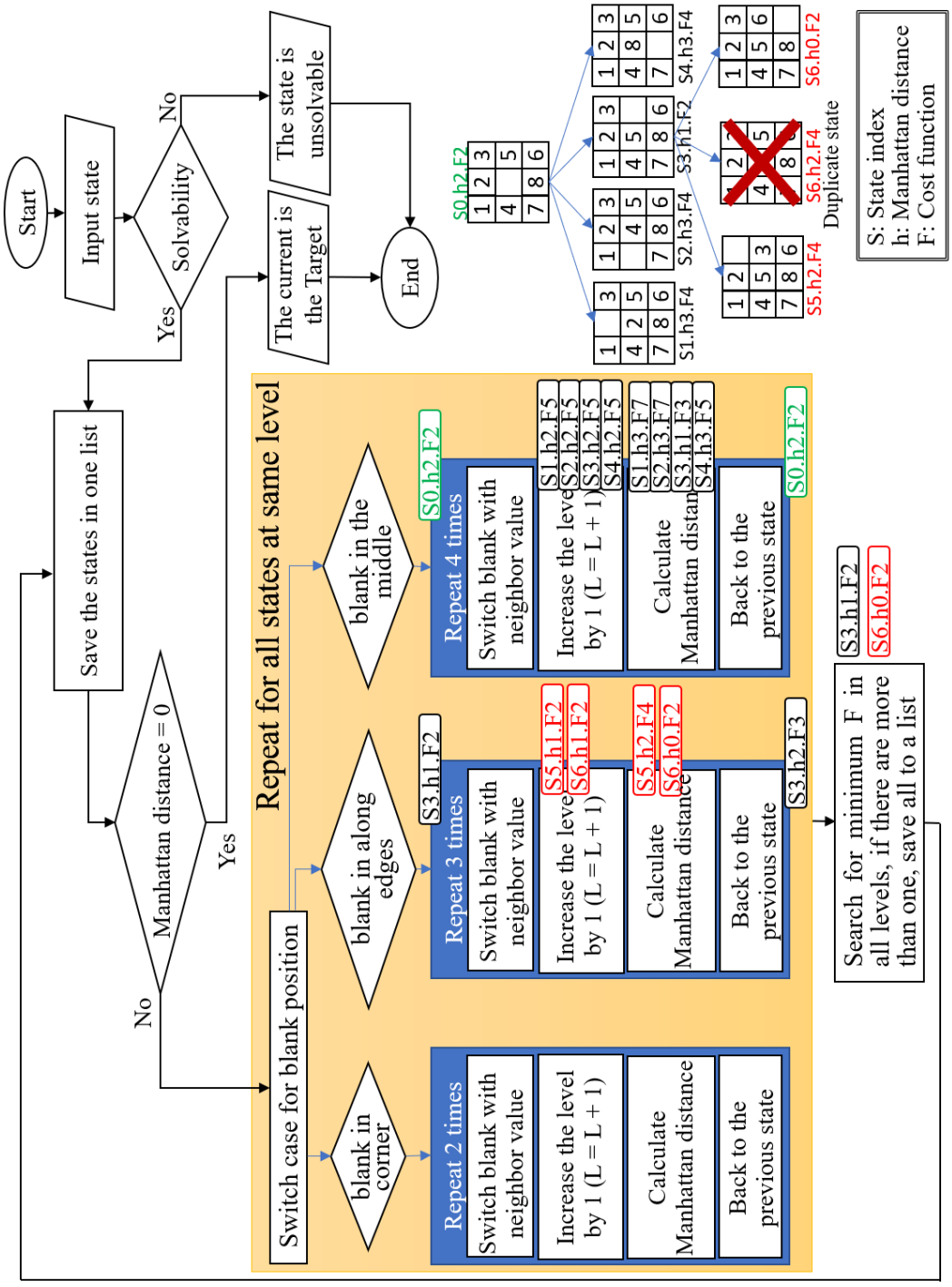


Figure 2.9. The implementation of the A-star algorithm for the 8-puzzle.

The conditional sentences in Algorithm 1 describe the implementation of the A* algorithm.

Algorithm: A* Implementation for 8-puzzle

```
1: if solvable then
2:    Check Manhattan distance
3: else
4:  End Algorithm
5: Repeat until finding the target
6:    if Manhattan ≠ 0 then
7:       Find a blank
8:       Perform Procedure switching blank
9:      Search for minimum cost
10:   else
11:      Input is the target
12:   end if
13: end repeat
```

The procedure of switching the blank with neighbors to generate branch nodes is described as follows:

Procedure: Switching blank

```
1:  if blank in a corner then
2:    Repeat 2 times: switch blank[1]
3:  else
4:  if blank in along edges then
5:    Repeat 3 times: switch blank[1]
6:  else
7:  if blank in the middle then
8:    Repeat 4 times: switch blank[1]
9:  end if
```

Switch blank contains 3 steps:

- Switch blank with a neighbor;
- Increase the depth (level in the tree which denotes the solution steps) by 1; and
- Recalculate Manhattan distance.

2.4 Solvability Condition

The solvability can be checked by the inversion, which indicates that a pair of tiles in the current state is in reverse order of their places in the goal state. Moving tiles in the puzzle horizontally doesn't affect the inversion, but, moving tiles vertically either increases the inversion by 2, decreases the inversion by 2, or doesn't change the inversion. Therefore, when the number of inversions is even, the puzzle is solvable; otherwise, it is unsolvable [38]. For example, if we have an 8-puzzle with the following configuration state (2, 1, 5; 4, blank, 3; 8, 6, 7), regardless of the blank, the inversion is calculated as follows:

The Investigated Tile	**Tiles Follow the Investigated Tile**	**Number of Inversions**
2	1	1
1	-	0
5	4 and 3	2
4	3	1
3	-	0
8	6 and 7	2
6	-	0
7	-	0
	Total inversions	6

The total inversions are six, which is an even number. Thus, the example configuration is solvable.

The solvability condition came up with a second problem in the algorithm which should be considered for the sequencing application, the unsolvable states. For unsolvable states, the sequencing system will be stuck and we would not be able to proceed in the sequencing process.

2.4.1 Proposal for the Solvability Problem

As mentioned before, the 8-puzzle has 9! different configurations, and only half of them are solvable. Since the state configurations in practical implementation in the warehouse are random, we will not be able to carry out sorting for unsolvable states (9! / 2 states in the case of the 8-puzzle). Therefore, we need a scenario in which all states of the puzzle are solvable. In order to build such a scenario, we provided a pre-sorting strategy.

The products moving to the sorting area enter in a random configuration, which might be an unsolvable configuration. Therefore, we have to pre-sort the products on the sequencing board so that the pre-sorted configuration is a solvable one. The pre-sorting process is as follows:

- Check the solvability by calculating the inversion number;
- In case of an odd number of inversions, move the first six tiles to their specific positions on the sequencing board; and
- Switch the last two tiles on the board.

Figure 2.10 shows a flowchart of the pre-sorting process, and Figure 2.11 shows an example of the pre-sorting process for an unsolvable input configuration.

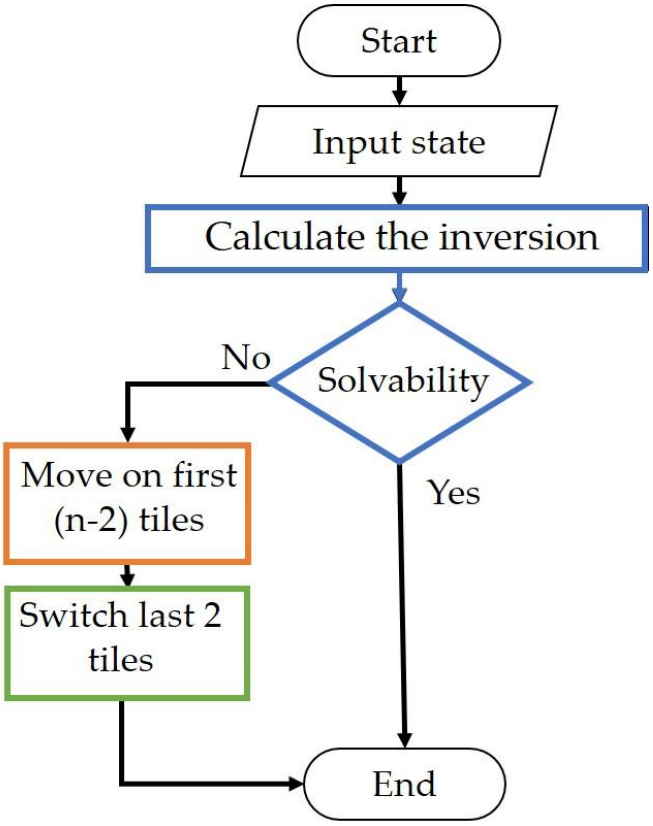


Figure 2.10. Flowchart of pre-sorting process.

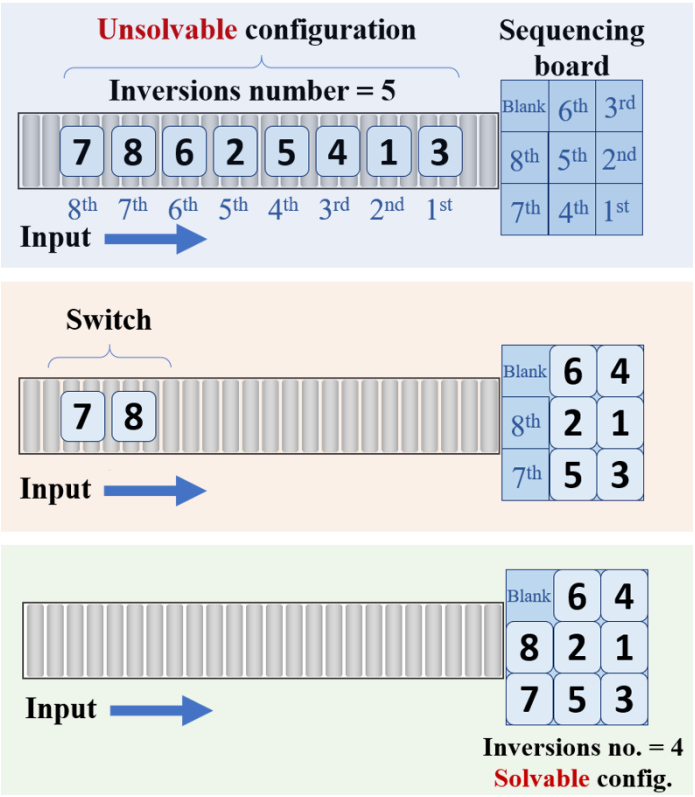


Figure 2.11. Pre-sorting process.

Applying the pre-sorting strategy, we could be able to solve all configurations of the puzzle.

Figure 2.12 illustrates the modified A-star algorithm used for our sequencing system.

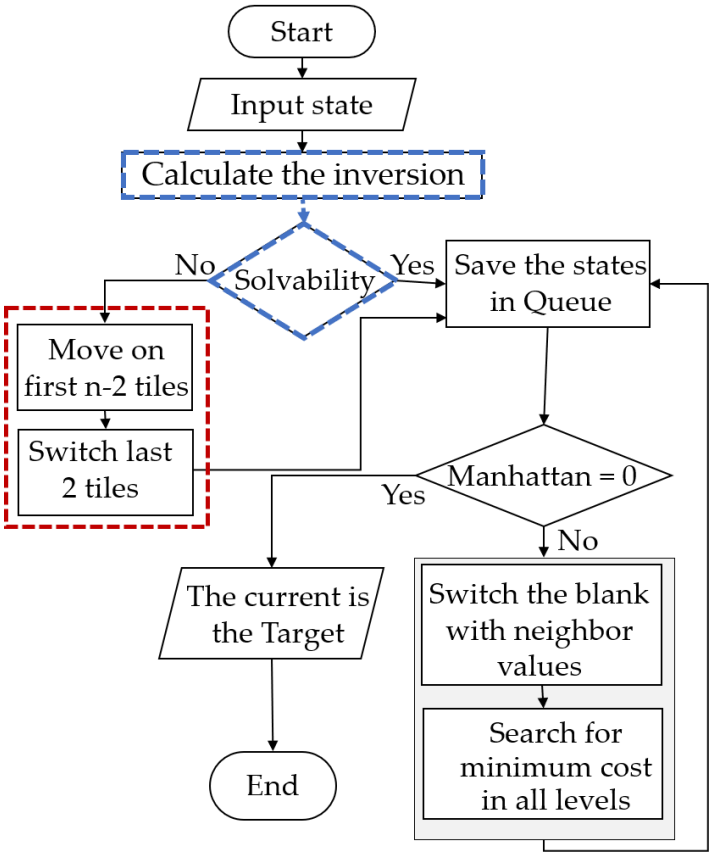


Figure 2.12. Modified A-star algorithm for n-puzzle.

The A-star algorithm is ready now to be implemented for the sequencing system.

2.5 Sequencing System Design

In the practical implementation in the warehouses, different parameters should be considered to choose the optimum sequencing board shape, size,

and number. Furthermore, the sequencing strategy for a different number of boxes.

2.5.1 Board Shape

Different shaped boards can carry out the sequencing task. Therefore, four different sizes with two shapes were discussed with the same number of tiles. A 2 × 3 puzzle has 6! = 720 states, and half of them are unsolvable. By keeping the blanks in the corner of the puzzle to satisfy the reality of practical implementation in the warehouse, we reduced this to only 60 solvable states. For the same configuration in both initial and goal states as shown in the example in Figure 2.13.

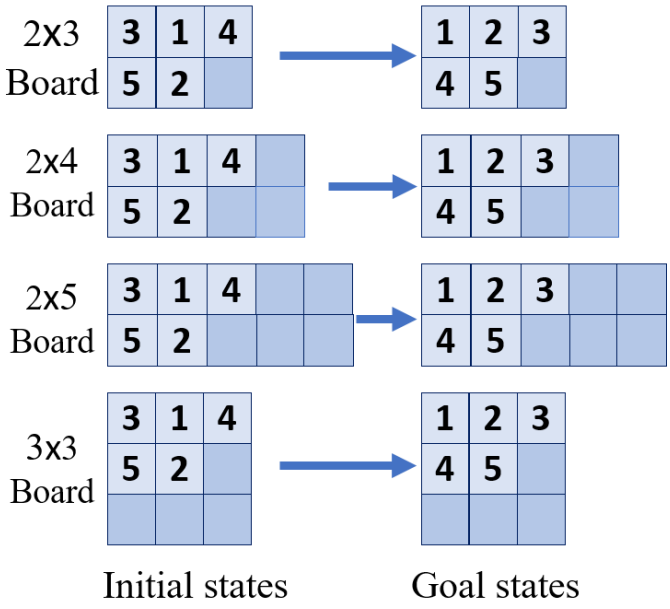


Figure 2.13. An example of examination same state configurations with different board sizes and shapes.

Figure 2.13 illustrates the effect of different board shapes and sizes of the puzzle on the solution steps for all 60 states. The results of Figure 2.14 are summarized in Table 2.1.

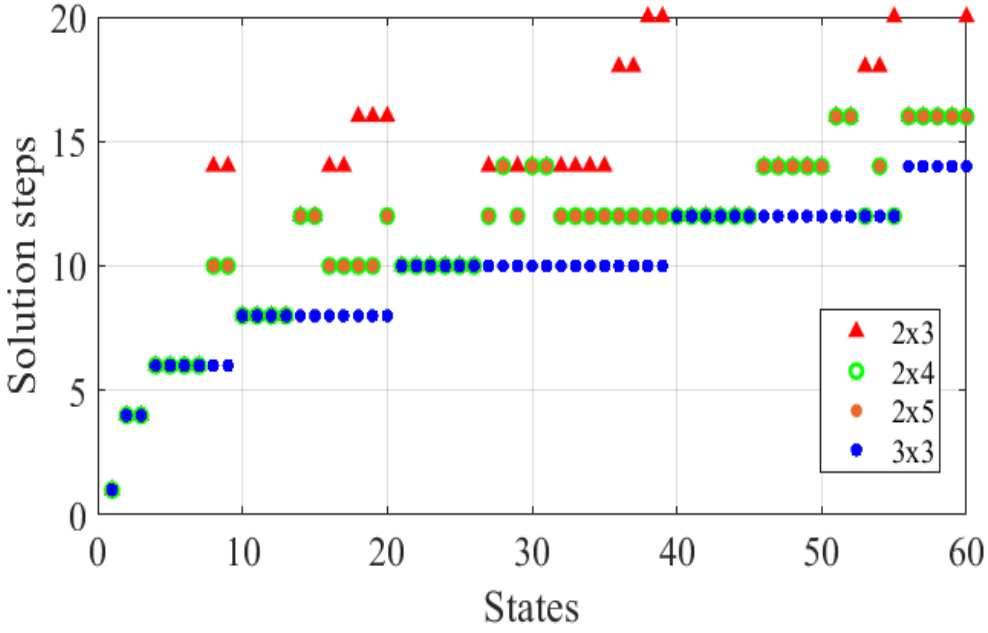


Figure 2.14. Comparison between different board sizes and shapes for the same number of boxes.

Table 2.1. Comparison of the performances of a 3 × 3 puzzle with different board sizes and shapes regarding the solution steps.

3 × 3	Better [%]	Same [%]	Worse [%]
vs. 2 × 3	61.6	38.4	0
vs. 2 × 4	58.3	41.7	0
vs. 2 × 5	58.3	41.7	0

From the table, the 3 × 3 board showed a better performance than the 2 × 3, 2 × 4, and 2 × 5 boards by 61.6%, 58.3%, and 58.3%, respectively. One of the reasons for these results is the difference in the number of blanks in the different shapes and sizes of the puzzle.

More analyses are necessary to verify the effectiveness of other factors on the overall solution steps for different shapes. For different shapes of the puzzle, there are many factors affect the overall solution steps such as branching factor, rectilinear distance, and the Aspect Ratio (AR) of the puzzle.

2.5.1.1 Branching Factor

The branching factor is the number of states that can be generated from each state in the tree. Usually, the branching factor measures the space complexity of the searching algorithm. The higher the branching factor, the lower the overhead of the repeatedly expanded states [35]. In our case, the analyzed data were generated from the target state, where we used the opposite concept of the branching factor. If the branching factor is higher, more states would be generated for a specific level in the tree (the level denotes the solution steps). Figure 2.15 illustrates an example of the effect of the branching factor on the number of generated states at the same level in the tree.

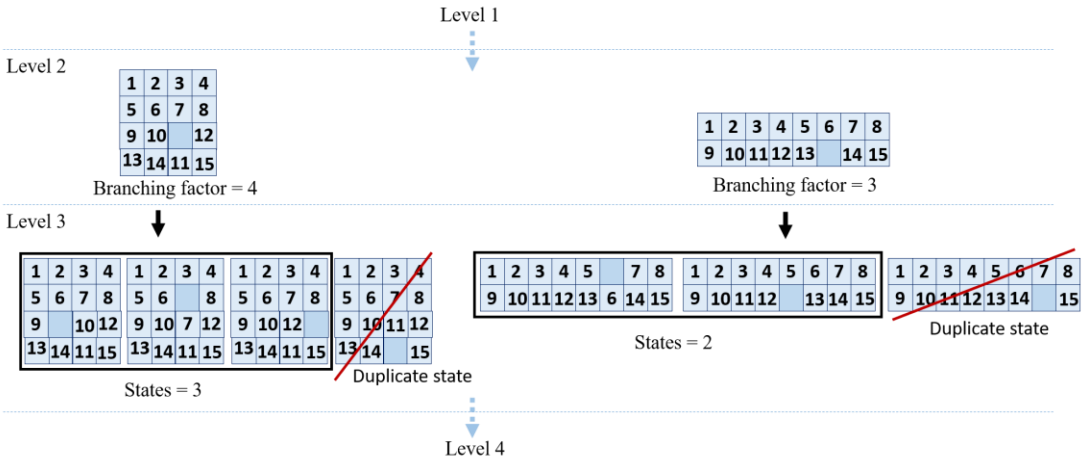


Figure 2.15. The effect of the branching factor on the number of generated states in the same level.

In Figure 2.15, two different shapes are illustrated, and we note that in level 3 (three steps to the solution), the square shape had more generated states than the rectangular one due to the difference in the branching factor. Figure 2.16 shows the average branching factor for both shapes discussed in the previous example.

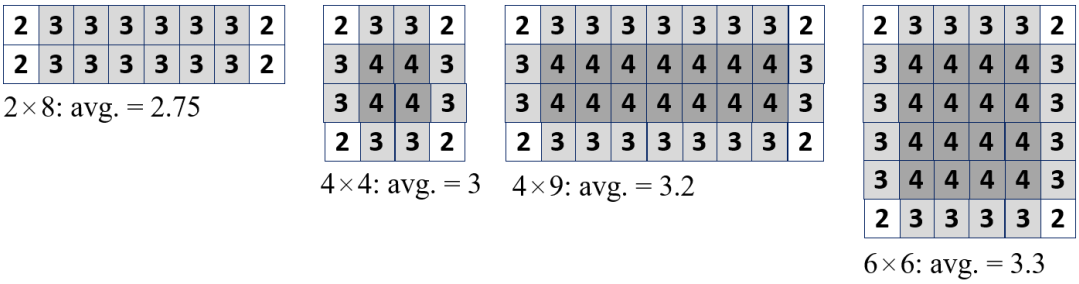


Figure 2.16. The average branching factor for different sizes and shape boards.

2.5.1.2 Maximum Rectilinear Distance of One Tile

We suggest Equation (2.3) for calculating the maximum steps of a tile:

$$\mathbf{r_d = (L + W) - 2,} \tag{2.3}$$

where r_d is the maximum rectilinear distance of the tile; L is the length of the board; and W is the width of the board.

A smaller distance for one tile results in a better board since it decreases the number of initial steps of the pre-sorting process. Figure 2.17 illustrates the maximum distance that the tile can move.

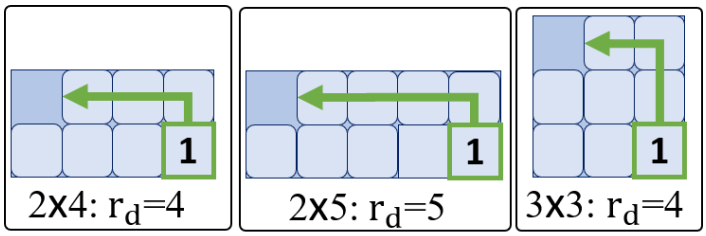


Figure 2.17. Maximum rectilinear distance of one tile of different board shapes and sizes.

From Figure 2.17, we noted that different board shapes could have the same r_d. With this in mind, we compared the performance depending on the maximum board capacity, as illustrated in Table 2.2.

Table 2.2. Comparison of different board shapes and sizes of puzzles, and the max. capacity in the case of the same $\mathbf{r_d}$.

Max. Rectilinear Distance of One Tile	Max. Capacity	Board Size
4	7	2 × 4
	8	3 × 3
5	9	2 × 5
	11	3 × 4
6	14	3 × 5
	15	4 × 4
7	17	3 × 6
	19	4 × 5
8	20	3 × 7
	23	4 × 6
	24	5 × 5
9	23	3 × 8
	27	4 × 7
10	26	3 × 9
	19	4 × 5
	35	6 × 6

From Table 2.2, we concluded that in the case of r_d, being the same for different board sizes and shapes, square puzzles provide more capacity than rectangular ones.

2.5.1.3 Pre-sorting Steps

The pre-sorting process plays a key role in the whole sorting system in practical implementations.

As mentioned in Section 2.6.2, the puzzle shape affects the rectilinear distance of one tile, r_d as well as the number of initial steps in pre-sorting. Figure 2.18 illustrates the initial steps to fill in the sequencing board with different sizes and shapes concerning r_d.

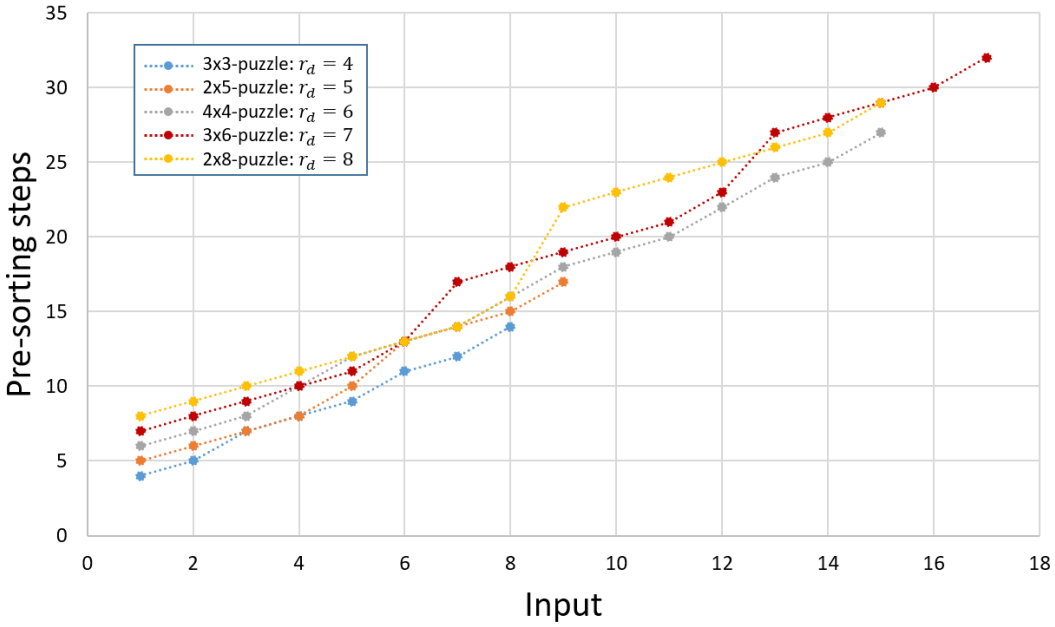


Figure 2.18. Effect of r_d on pre-sorting steps.

As is clear from Figure 2.18, increasing the rectilinear distance of one tile will also increase the pre-sorting steps. However, a reasonable question arises when dealing with different shapes: how does the Aspect Ratio (AR) of the puzzle affect the performance in terms of solution steps? To answer this question, we investigated the relationship between the aspect ratio and rectilinear distance.

2.5.1.4 Aspect Ratio

The Aspect Ratio is the number of columns divided by the number of rows of the puzzle, and this has a direct effect on the rectilinear distance of one tile, r_d, and further on the pre-sorting steps. Table 2.3 illustrates the corresponding r_d of the aspect ratio for the different puzzle shapes and sizes outlined previously.

Table 2.3. Aspect Ratio and rectilinear distance of one tile for different puzzle sizes.

Puzzle size	Aspect Ratio	Rectilinear distance
4 × 4	1	6
2 × 8	4	8
6 × 6	1	10
4 × 9	2.25	11
3 × 12	4	13
2 × 18	9	18

According to Table 2.3, we confirmed the direct relationship between the aspect ratio and rectilinear distance. Thus, a smaller AR reduces the r_d, which also reduces the pre-sorting steps.

If we considered preliminary that a square puzzle is better than a rectangular one, we have to investigate the puzzle size.

2.5.2 Board Size and Number

In real-world warehouses, under the batched/or zoning picking policy, the retrieved items from the storage area necessitate being either consolidated or sequenced in the way of satisfying the order sequence by customers. The number of these items is very varied depending on the order lists. In [12], the picking method was to accumulate the orders in separated bins under batched/or zoning picking policy. In this case, the orders would be released from the bins, then re-sequenced in the desired sequence. Boysen et al. generated two differently sized datasets for their computational study, a small instance that involves 12 orders in 24 bins, while the large instance involves 20 orders in 40 bins which are of a real-world size. These values are chosen based on practitioners' information. The number of the boxes needed to be

sequencing at the same time is difficult to be determined because it is depending on several factors such as the boxes sizes, boxes weights, and pallets capacities and sizes. Therefore, in practical implementation, an 8-puzzle board would face a limitation by restrictions of the maximum capacity of the puzzle (8 boxes). In such a case, we propose and discuss three strategies for sequencing more than 8 items.

2.5.2.1 Increase the Size of the Sequencing Board

To carry out the sequencing process for more than 8 boxes, bigger board sizes can be used. Since it was concluded preliminary that the square shape provided better performance than the rectangular one, we used in this strategy an extension version of the 8-puzzle which is the ($n^2 - 1$) puzzle.

Figure 2.19 illustrates the strategy of using 15-puzzle as an example for practical implementation.

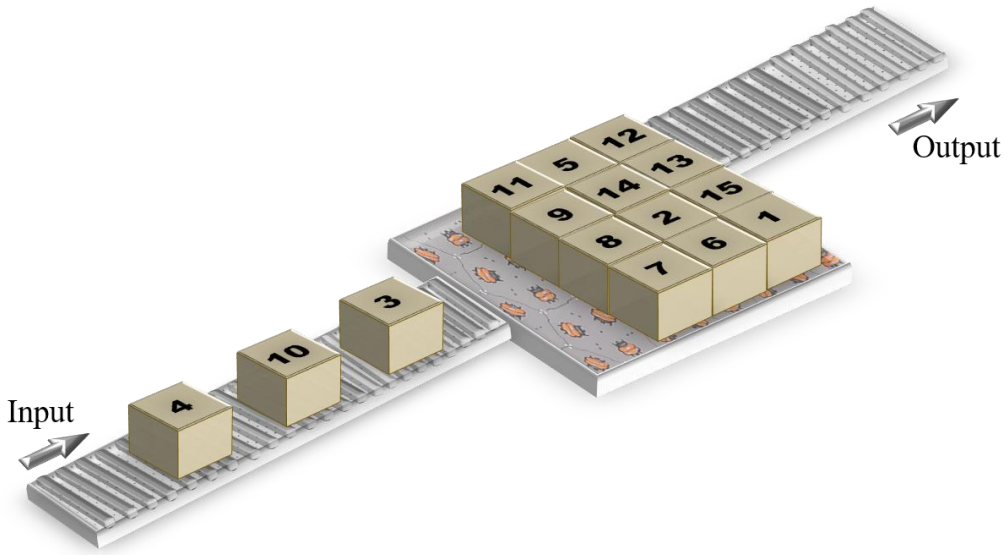


Figure 2.19. The strategy of using 15-puzzle for practical implementation.

As in figure 2.19, the main input conveyor feeds the sequencing board. After filling all the incoming boxes on the board, the sequencing process will start.

In this study we considered the sequencing time as a function of steps, therefore we calculated the pre-sorting steps and the puzzle solution steps.

Two main factors were considered, the area occupied by the system regardless of the main input conveyor, and the sequencing time which is the solution steps until we get the final goal of the desired sequence of boxes.

I. Area for the Strategy of Different Sizes of Board

The area occupied by the system can be calculated as follow:

$$\boldsymbol{A = (C_b + 1) \times A_{box}} \tag{2.4}$$

where A is the total area occupied by the system regardless of the main input conveyor; C_b is the maximum capacity of the board (15 boxes for 15-puzzle); and A_{box} is the box' area.

As we can see in Equation (2.4), the number of boxes does not affect the area as long as $\boldsymbol{N \leq C_b}$. where N is the input (number of boxes).

II. Time for the Strategy of Different Sizes of Board

We considered the time as a function of solution steps. we have considered the pre-sorting steps and the sequencing steps on the board, ignoring the moving boxes on the main feeding conveyor.

The total sequencing steps are calculated as Equation (2.5):

$$\boldsymbol{S_{total} = \begin{cases} S_n + S_{p\_n} & if\ N < C_b \\ S_{max} + S_{p.max} & if\ N = C_b \end{cases}} \tag{2.5}$$

where $\boldsymbol{S_{total}}$ is the total number of steps; $\boldsymbol{S_n}$ is the Solution steps for n boxes; $\boldsymbol{S_{max}}$ is the Maximum steps to solve the puzzle; $\boldsymbol{S_{p\_n}}$ is the Pre-sorting steps (the initial steps to fill in the board with n boxes); $\boldsymbol{S_{p.max}}$ is the Pre-sorting steps (the initial steps to fill in the board with full capacity); N is the input (number of boxes); C_b is the maximum capacity of the board.

Table 2.4 illustrates the maximum number of solution steps for 15-puzzle, 24-puzzle, 35-puzzle, and 48-puzzle, and the board area as a function of the box size.

Table 2.4. Maximum solution steps for 15, 24, 35, 48 puzzles, and the board area [39].

Puzzle capacity	**15**	**24**	**35**	**48**
Maximum capacity	80	205	405	716
Area [box size]	16	25	36	49

As shown in the table, the maximum solution steps are drastically increased by increasing the board size. Furthermore, the system is still restricted by the limitation of the puzzle capacity.

2.5.2.2 Using Multi-Boards

As mentioned in 2.7.1, increasing the board size could carry out the sequencing system for more than 8 boxes. However, such a strategy is still restricted by the maximum capacity of the puzzle.

In this strategy, we used the same size of the puzzle with an increase in the number of boards as shown in figure 2.20.

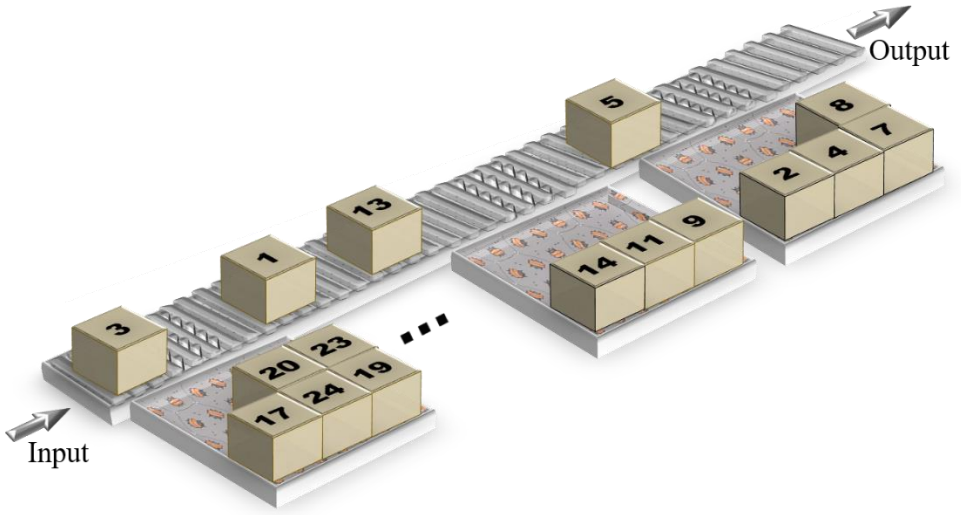


Figure 2.20. Several boards along with the input line for the sorting process.

As shown in figure 2.20, we placed the boards along with the input conveyor. The boxes coming in random sequence are separated into these boards based on their identification numbers (IDs) (ex. The boxes from 1 to 8 enter the first board, the boxes from 9 to 16 enter the second board, and so on).

We assumed that when the boxes are entering the boards, they can move simultaneously, and the sequencing process will start after filling in the boards with their assigned boxes.

Keeping these assumptions in mind, the sequencing process will be carried out in parallel in all boards, doing so allowing to reduce the waiting time if the sequencing will be carried out in series.

We investigated the same two factors as 2.5.2.1, which are the area and the time.

I. Area for the Strategy of using Multi-Boards

The area used by the system in this strategy is calculated as Equation (2.6):

$$\boldsymbol{A = [C_b + 1)N_b] \times A_{box}} \tag{2.6}$$

where A is the total area occupied by the system regardless of the main input conveyor; C_b is the maximum capacity of the board (8 boxes for 8-puzzle as the

example in figure 5.2); A_{box} is the box' area; and $\boldsymbol{N_b}$ is the number of boards which can be calculated as follow:

$$\boldsymbol{N_b} = \left\lceil \frac{\boldsymbol{N}}{\boldsymbol{C_b}} \right\rceil \tag{2.7}$$

where N is the input (number of boxes).

We verified from Equation 2.6 and Equation 2.7 the direct relationship between the number boxes and the area of the system.

II. Time for the Strategy of using Multi-Boards

As in the strategy of different sizes of board, we considered the time as a function of solution steps.

In this strategy, the boxes are moving on the boards simultaneously and the sequencing process starts after filling all the boards. We calculated the time in the worst case as in Equation 2.8.

$$\boldsymbol{S_{total}} = \boldsymbol{S_{max}} + (\boldsymbol{N_b} - \mathbf{1})\, \boldsymbol{S_{p.max}} + \boldsymbol{S_{p\_r}} \tag{2.8}$$

where $\boldsymbol{S_{total}}$ is the total number of steps; $\boldsymbol{S_{max}}$ is the Maximum steps to solve the puzzle; $\boldsymbol{N_b}$ is the number of boards; $\boldsymbol{S_{p.max}}$ is the Pre-sorting steps (the initial steps to fill in the board with full capacity); $\boldsymbol{S_{p\_r}}$ is the Pre-sorting steps (the initial steps to fill in the board for remaining boxes).

The remaining boxes can be calculated as follows: $\mathbf{p\_r} = \boldsymbol{N} - (\boldsymbol{N_b} - \mathbf{1})\, \boldsymbol{C_b}$.

In Equation 2.8, we considered the $\boldsymbol{S_{max}}$ only one time for all boards, the reason is that all boards work in parallel, and in the worst case, at least one of them needs the maximum solution steps.

2.5.2.3 Adding a Buffer Line

In this strategy, fixed board size is used and the system has been extended with a buffer conveyor used to store temporarily the boxes. The buffer line is fixed along with the main input conveyor as shown in figure 2.21.

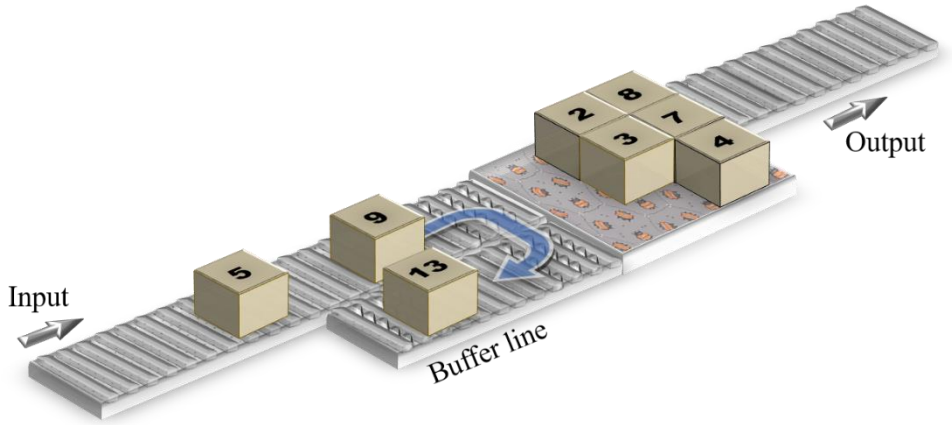


Figure 2.21. Adding buffer line along with the input conveyor for the sorting process.

As shown in figure 2.21, the buffer line is placed along with the main input conveyor in a way the boxes can be buffered and reenter again to the main conveyor.

For 8-puzzle board size, when the boxes arrived at the board, if the box is from 1 to 8 it will enter the board, otherwise, it will move left to store temporally in the buffer line. After filling in the board with 8 boxes, the first sequencing process starts. Afterward, the remaining boxes will be released from the output point of the buffer line to the main conveyor. Again, the boxes from 9 to 16 will enter the board (after releasing the sequenced boxes in the first sequencing process), and the remaining boxes will store in the buffer line.

In this strategy, the sequencing processes are carried out in series, board after board. As for other strategies, we investigated both area and time for this strategy.

I. Area for the Strategy of Adding Buffer Line

The area for this strategy is calculated as Equation 2.9.

$$\boldsymbol{A = (N + 1) \times A_{box}} \quad (2.9)$$

where A is the total area occupied by the system regardless of the main input conveyor; and A_{box} is the box' area.

As it is clear from Equation 2.9, unlike the strategy of using multi-boards, the area here is depending only on the number of boxes.

II. Time for the Strategy of Adding Buffer Line

The same analyses of previous strategies to calculate the sequencing time was carried out in this strategy. In this strategy, the boxes are sequenced on the board based on the used puzzle capacity. Therefore, for boxes more than the puzzle capacity, the remaining boxes would wait for the next sequencing process. In this waiting time, the first sequencing process is carried out.

We calculated the time in the worst case as in Equation 2.10.

$$\boldsymbol{S_{total} = (N_b - 1)\, S_{max} + S_r + (N_b - 1) S_{p.max} + S_{p\_r} + S_{buffer}} \quad (2.10)$$

where $\boldsymbol{S_{total}}$ is the total number of steps; $\boldsymbol{S_{max}}$ is the Maximum steps to solve the puzzle; $\boldsymbol{S_r}$ is the steps to solve the puzzle for remaining boxes; $\boldsymbol{N_b}$ is the number of boards which refer to the number of sequencing processing times; $\boldsymbol{S_{p.max}}$ is the Pre-sorting steps (the initial steps to fill in the board with full capacity); $\boldsymbol{S_{p\_r}}$ is the Pre-sorting steps (the initial steps to fill in the board for remaining boxes); and $\boldsymbol{S_{buffer}}$ is the steps or buffered boxes for the next sequencing process. The remaining boxes can be calculated as follows: $\mathbf{p\_r} = \boldsymbol{N - (N_b - 1)\, C_b}$.

We assumed that one step is for entering the buffer line, and one step for outgoing to the main conveyor, Thus, each buffered box needs 2 steps. we assumed also that moving boxes on the main conveyor and the buffer-

conveyor are not calculated. Keep these assumptions in mind, $\boldsymbol{S_{buffer}}$ can be calculated in the worst case as follow:

$$\boldsymbol{S_{buffer} = \sum_{i=1}^{N_b-1} 2N - (i \times 2C_b)} \quad (2.11)$$

where N is the input (number of boxes); and $\boldsymbol{N_s}$ is the number of boards which can be calculated as follow: $\boldsymbol{N_b} = \left\lceil \frac{N}{C_b} \right\rceil$. where C_b is the maximum capacity of the used board.

The results of these strategies will be given in Chapter 3.

According to the literature illustrated in Chapter 1, the systems designed for storing and retrieval items that depended on the puzzle movement concept investigated the item retrieval time with multiple escorts (escort refers to the empty space in the puzzle system).

by increasing the number of escorts, the retrieval moves are decreased and the retrieval time is decreased as well [7, 27]. The analyses in these researches were conducted to retrieve an item to a specific point in the puzzle.

However, in Sam Loyd's puzzle which consists of (n × m) − 1 square tile and one blank, the concept is to keep sliding the blank in one of four cardinal directions until we reach the goal state, in such a puzzle system, all items should move to their position as in goal configuration. In this study, we investigate the increase of blanks on the solution steps for Sam Loyd's puzzle which is used in our sequencing system.

2.5.3 Number of Blanks

There are lots of researchers who worked on the sliding puzzles [29, 30, 32, 33, 40–42]. However, To the author's best knowledge, none of these literatures has addressed the case of a different number of blanks. Unlike other researches,

we first investigated the effect of one and two blanks on the solution steps for different sizes of the puzzle. Table 2.5 illustrates the effect of one and two blanks on different puzzle sizes regarding the maximum puzzle capacity, the total number of states, the number of solvable states, and the maximum number of solution steps.

Table 2.5. The effect of one and two blanks on different puzzle sizes.

	2×2 puzzle		**2×3 puzzle**		**2×4 puzzle**		**3×3 puzzle**	
No. of blanks	1	2	1	2	1	2	1	2
Max. capacity	3	2	5	4	7	6	8	7
No. of states	24	12	720	360	40,320	20,160	362,880	181,440
solvable states	12	12	360	360	20,160	20,160	181,440	181,440
Max. solution steps	6	4	21	12	36	26	31	24

According to the table, by increasing the number of blanks in the puzzle, the maximum solution steps is decreasing.

The second analysis is to show the effect of an arbitrary number of blanks on the solution steps for the 8-puzzle as illustrated in table 2.6.

Table 2.6. The effect of an arbitrary number of blanks on the solution steps for 8-puzzle

Number of blanks	**Maximum capacity**	**Maximum states**	**Maximum solution steps**	**Average solution steps**
1	8	362,880	31	21.97
2	7	181,440	24	16.03
3	6	60,480	21	12.7
4	5	15,120	17	9.99
5	4	3,024	13	7.88
6	3	504	10	5.93
7	2	72	7	3.8

As shown in Table 2.6, increasing the number of blanks will always decrease the solution steps. However, the increasing of blanks has the opposite effect

on the system which is decreasing the puzzle capacity. We noticed that the maximum number of blanks is 7, the puzzle, in this case, can sequence only two boxes.

Since we are dealing with multiple blanks in the puzzle, that allows us to carry out multiple steps simultaneously as double switching.

2.5.4 Double Switching

In the sliding puzzle, the step term denotes sliding the blank from its current position to the next position by switching it with the neighbor tile as shown in figure 2.22.

Figure 2.22. The concept of step in sliding puzzle.

If we assume that one step takes one time unit, means one switching carried out by one step and consumed one time unit. Keeping that in mind, we solve the puzzle with 2 blanks as in the example illustrated in figure 2.23.

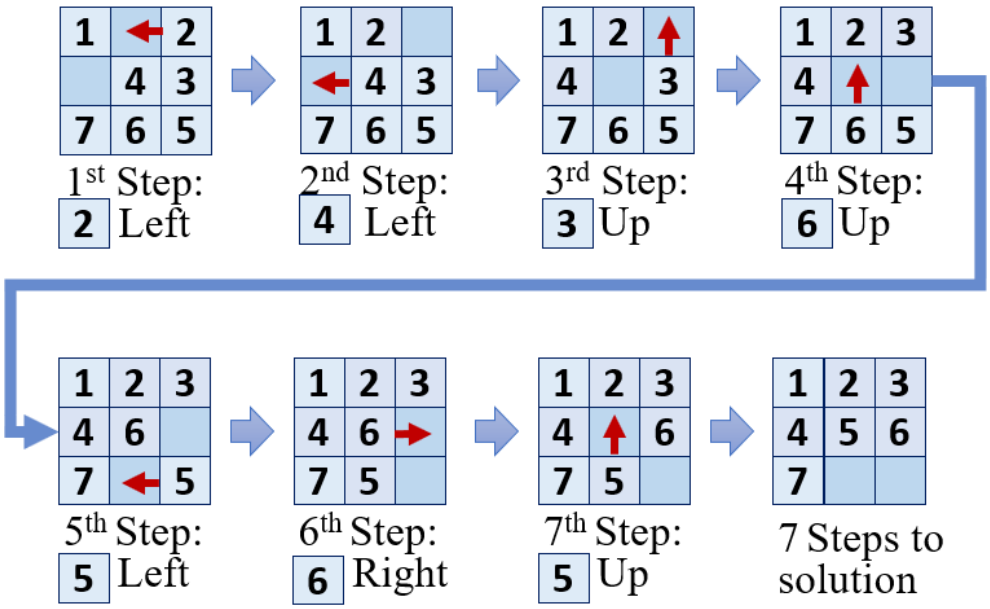


Figure 2.23. An example of one step concept to solve 8-puzzle with 2 blanks.

As shown in figure 2.23, this configuration takes 7 steps to reach the solution. We carried out 7 switchings and that consumed 7 times units. However, we noticed that the first step and the second step are independent steps. In such a case, we carried out switching the tiles (2 and 4) simultaneously. This double switching consumed one time unit, therefore, we considered it as one step. So, we can achieve double switching in one step.

By applying the new concept of double switching in one step to solve the previous example, we reduced the solution steps from 7 steps to 4 steps as shown in figure 2.24.

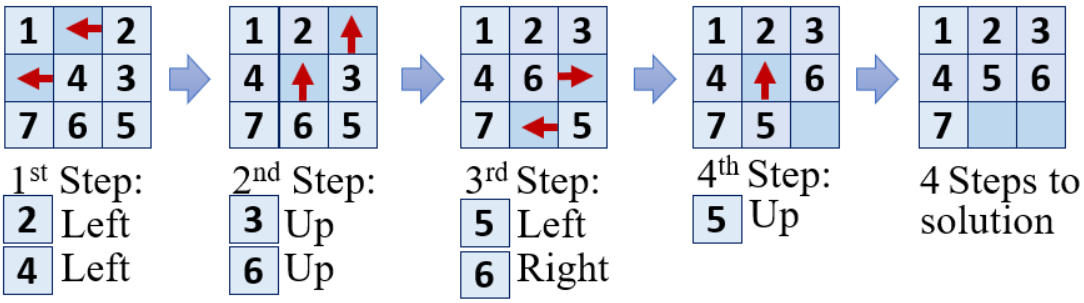


Figure 2.24. An example of the double switching concept to solve 8-puzzle with 2 blanks.

According to figures 2.23 and 2.24, double switching in one step concept can be applied only when the tiles are switched independently. We confirmed that each double switching can reduce the steps by 1 step, in the example we carried out the double switching 3 times, reducing the solution steps by 3 steps in total.

The analysis was carried out using "MATLAB 2020b"software as follows:

1. Searched the solution steps as one switching is carried out in one step.
2. Tracked the positions of the first and second blanks.
3. Assumed that each switching is assigned to one blank, and we can't carry out double switching for the same blank.
4. Checked independence of double switching.

Figure 2.25 shows an example of the double switching process in MATLAB.

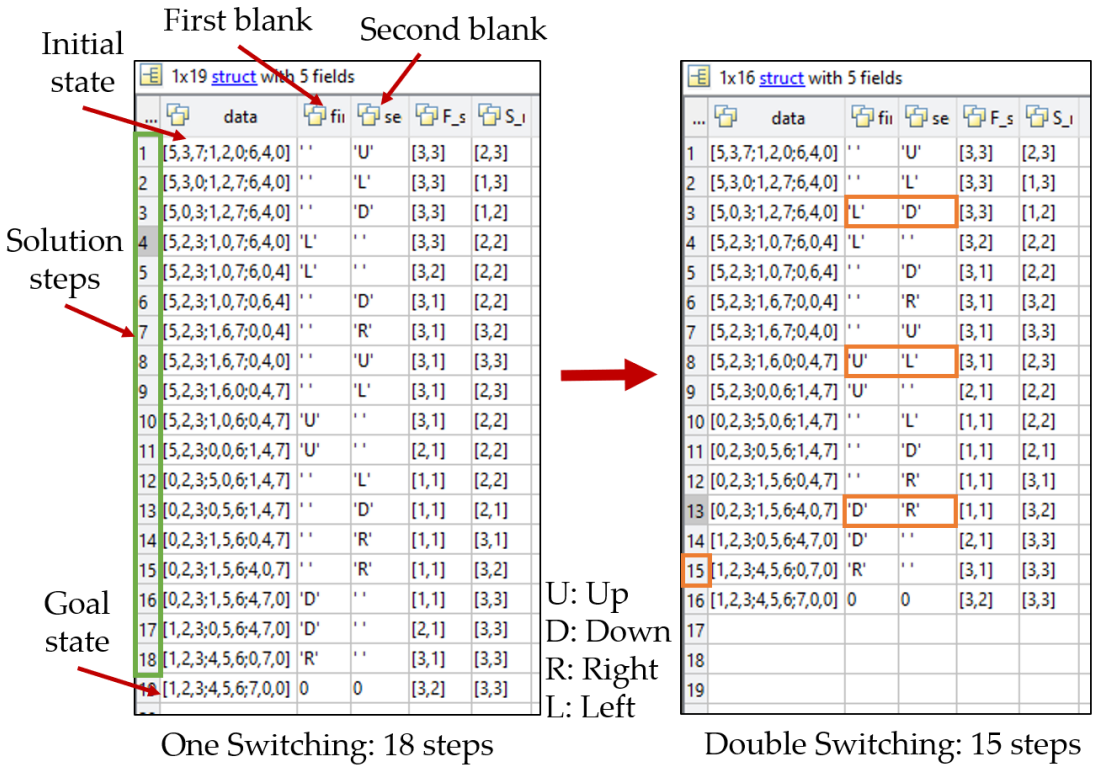


Figure 2.25. An example of the double switching process in MATLAB.

In the example, the initial state is [5, 3, 7; 1, 2, blank; 6, 4, blank]. At the programming level, we represent the blanks by zeros.

The analysis steps carried out by MATLAB are as follows:

1. Get the solution steps by applying the one switching in one step concept, in the example, the solution steps are 18 steps;
2. Confirm that each switching is assigned to a different blank by comparing the current state's blanks with the next state's blanks;
3. Check the independency of the blanks, which can be done by the concept of Hamming distance as follow:
   - Give the current state a Hamming = 0;
   - Check the Hamming of the current state with the state (current + 2);
   - If the Hamming = 2, the blanks are independent, else they are dependent.

In the example, the solution steps are 18 and 15 steps before applying the double switching and after, respectively. Thus, we reduced the solution steps by 3 steps.

Figure 2.26 illustrates the procedure algorithm to apply double switching.

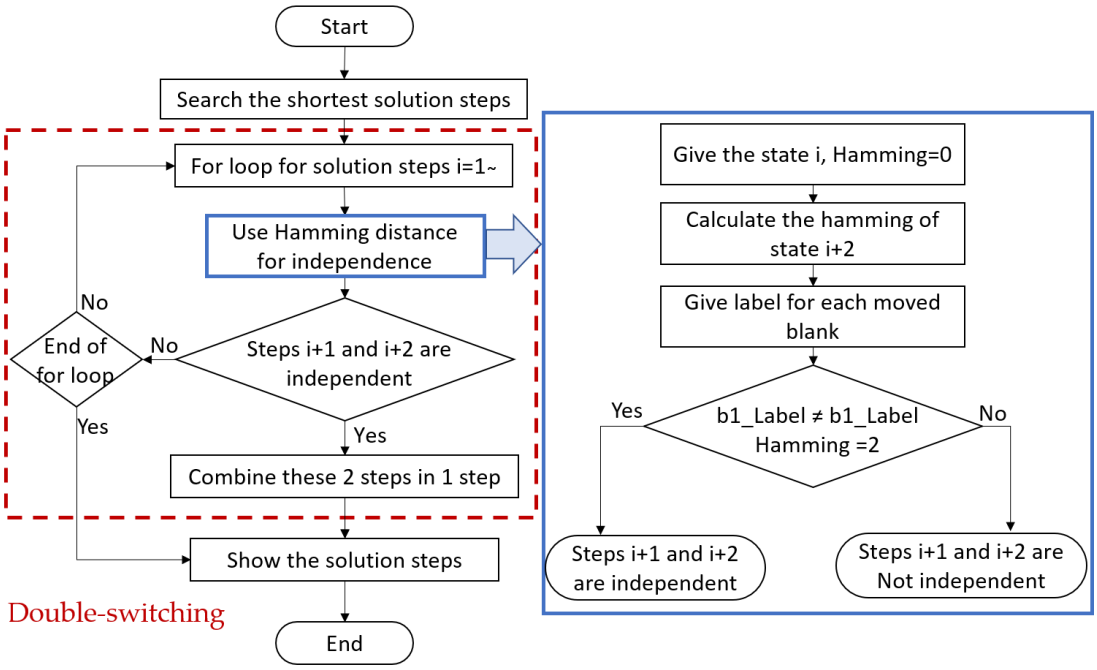


Figure 2.26. the procedure algorithm to apply the double switching process.

2.5.4.1 Improvements with Block Movement

In the example described in 2.5.4, we assumed that one switching should be applied for one blank and we can't carry out the second switching for the same blank. However, we noticed some cases where the same steps are applied for the same blank in the current state and the next state. For instance, in steps 3 and 4 in the example shown in figure 2.25 (right) after applying double switching, the first zero is moved left in both steps, in this case, we can apply the block movement concept.

In this concept we move the two tiles together as one block to the right, the blank would be moved to the left as well. To apply the block movement, we follow the following steps:

1. After applying the simple double switching, we check the cases of moving the same blank twice in the same column or the same row.
2. Confirm the independency of another blank with the blank we are applying block movement on. The blanks independency can be done by tracking the blanks locations as follows:
   - If the blank moves up or down in both current and next states, another blank in the next state and the state (current +2) should not be in the same column of the blank we are working on.
   - If the blank moves left or right in both current and (current +1) states, another blank in the (current +1) and (current +2) states should not be placed in the same row of the blank we are working on.

Figure 2.27 illustrates the procedure algorithm to apply double switching with block movement.

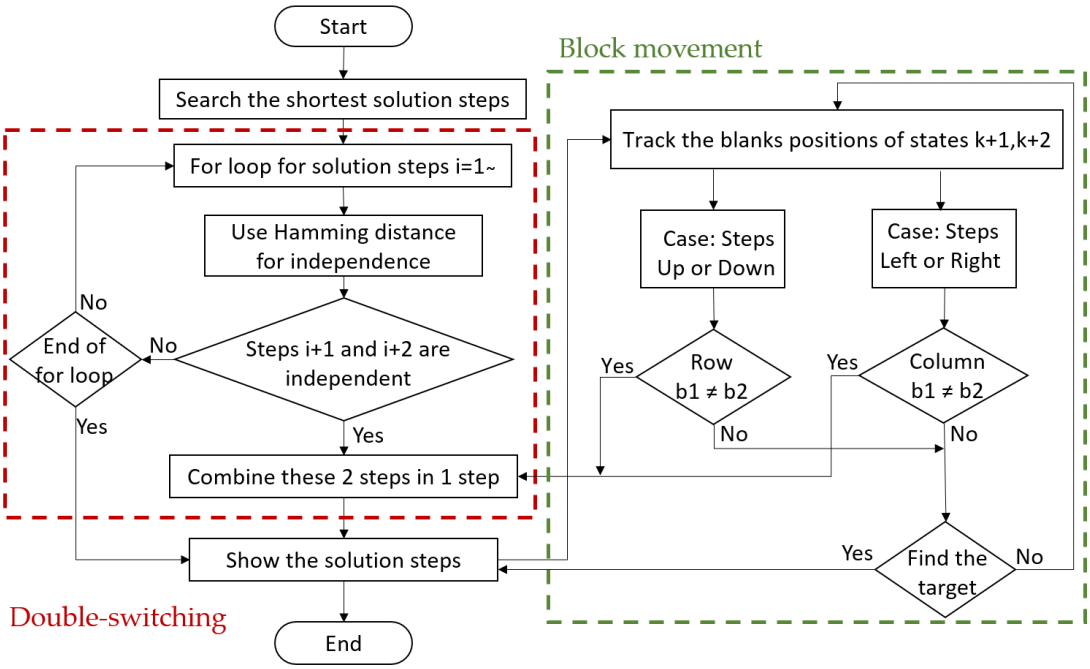


Figure 2.27. the procedure algorithm to apply the double switching process with block movement.

Figure 2.28 shows the steps of solving the same example in figure 2.25 after applying block movement.

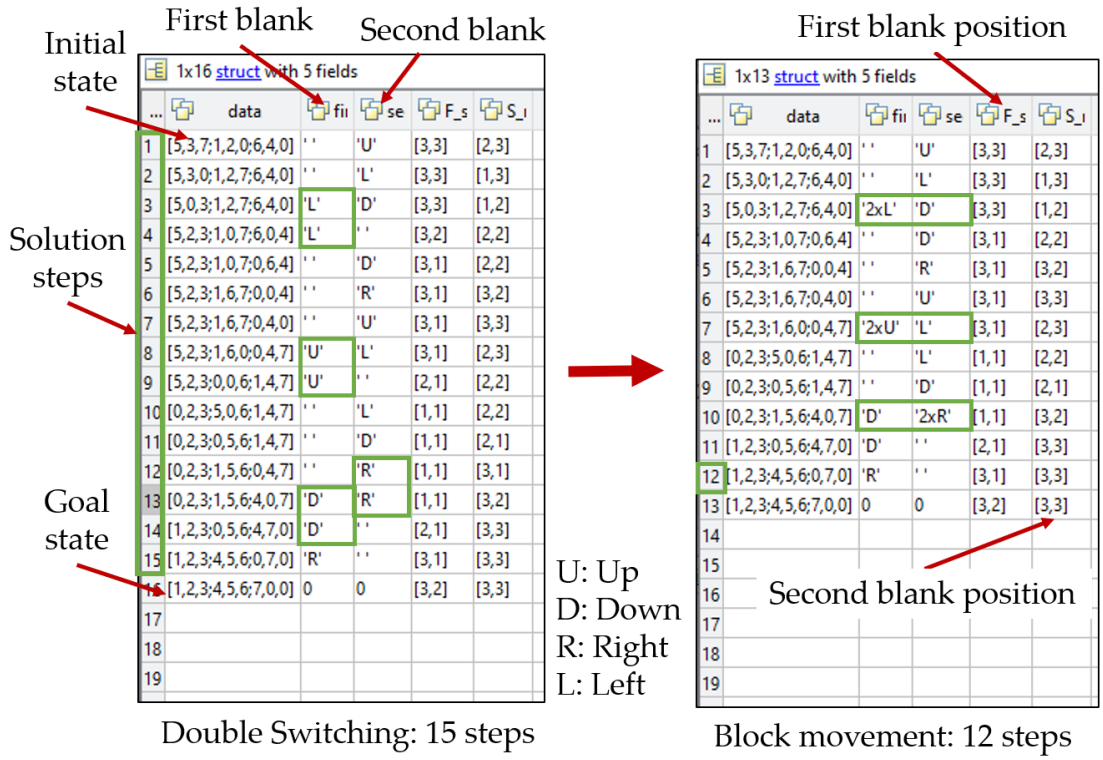


Figure 2.28. An example of applying block movement for an 8-puzzle with 2 blanks.

According to figure 2.28, the first blank had 3 cases where the same steps are applied in both current and next states, and one case for the second blank However, based on the condition described for the allowability of applying block movement, only 3 cases can allow the block movement.

In this example, we reduced the solution steps by 3 steps. By applying the double switching with block movement, we can carry out 3 steps simultaneously in one step as shown in figure 2.29.

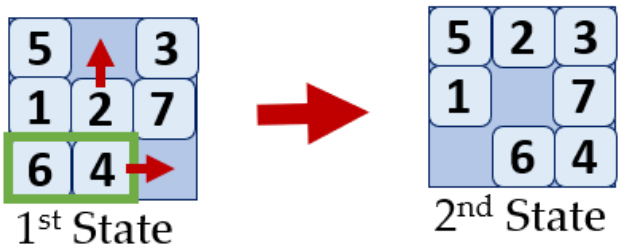


Figure 2.29. The concept of double switching with applying block movement.

2.6 Summary

In this chapter, we presented the puzzle-based system. We discussed two solving methods: the game tree and the path-finding algorithms. We chose the A-star algorithm which was optimally efficient among the algorithms that extend search paths from the root. We modified the A-star algorithm to fulfill the sequencing process by adding the solvability condition and pre-sorting strategy by switching the last (n-2) tiles in the puzzle in the case of unsolvable configuration which are 50% of puzzle permutations.

In this chapter, we investigated also the effect of several factors on the overall solution steps. For instance, branching factor the Aspect Ratio and the rectilinear distance.

To meet the practical implementation in real-world warehouses, we proposed three strategies, increasing the board size using different puzzle board sizes, using multi-boards, and adding buffer line. In addition, we dealt with the puzzle with an arbitrary number of blanks. Firstly, we investigated the effect of increasing the number of blanks in the puzzle on the maximum solution steps. Then, we investigated the effect of simultaneous double switching in one step on the system regarding maximum solution steps. By carrying out simultaneous double switching in 8-puzzle with 2 blanks, we would be able to reduce the maximum solution steps considering that one double switching can reduce the steps by 1 step.

CHAPTER 3

RESULTS AND DISCUSSION

After the overall explanation in the methodology chapter, here, we present the results of the numerical equations and we discuss these results regarding reducing the total number of steps keeping in the mind the usage area by the system.

The results and discussion will be presented in different suction following the same order in Chapter 2.

3.1 Puzzle Shape

We investigated in Chapter 2 different factors that affect the solution steps in both a square board and a rectangular one, the summary of these factors' effect is illustrated in table 3.1.

Table 3.1. Our generated tree for the 8-puzzle vs. other works.

Factor	Square puzzle board	Rectangular puzzle Board
Branching factor	✓	✗
Rectilinear distance	✓	✗
Presorting steps	✓	✗
Aspect ratio	✓	✗

In order to generalize the comparison of different shapes of the puzzle, the same size and number of blanks were used. First, we investigated the 16-boxes size of the puzzle.

3.1.1 16-boxes Size

This size can sort 15 boxes with two different shapes (4 × 4 and 2 × 8). As Figure 14 shows, we generated 2×10^5 non-random states for both shapes, starting from the target state. Figure 3.1 illustrates the performance of the generated states regarding the solution steps.

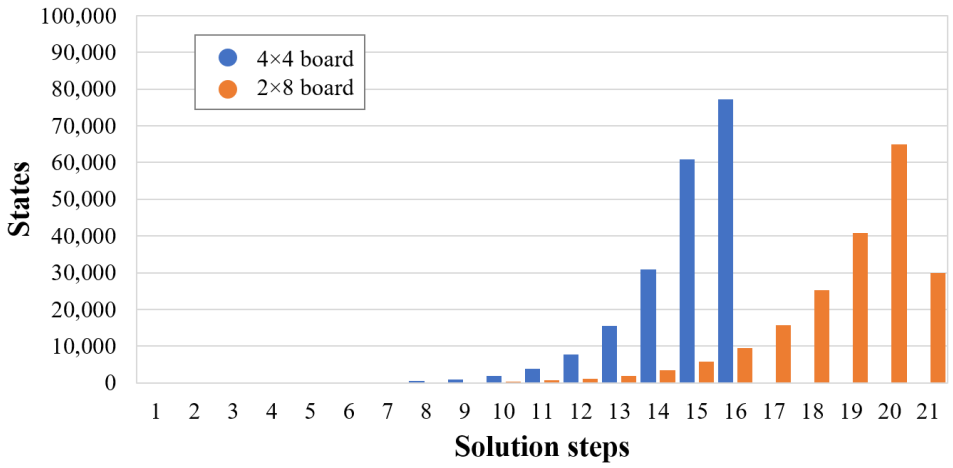


Figure 3.1. Comparison between the 4 × 4 and 2 × 8 board sizes with non-random states regarding the solution steps.

To verify the validity of our method to generate the state tree, we compared our 8-puzzle tree with other works regarding the following factors: maximum number of states, maximum solution steps, and average solution steps. Table 3.2 illustrates the comparison between our results for the 8-puzzle with others.

Table 3.2. Our generated tree for the 8-puzzle vs. other works.

Comparison Factor	Our Generated Tree	Other Works [29, 32, 43]
Maximum number of states	181,440	181,440
Maximum solution steps	31	31
Average solution steps	21.97	≈ 22

According to Table 3.2, we were able to validate our method, and the same program was used to generate the 2×10^5 states for different shapes in this section.

In the generated tree of the 16-boxes size of the puzzle, we noticed clearly a significant difference in the state numbers of the two puzzles in the same tree depth (solution steps). In other words, states in one shape of the puzzle need more solution steps than the second shape. The equation that describes the number of states that need more solution steps is as follows:

$$\boldsymbol{N} = \sum_{\mathbf{i}=\mathbf{S_{max.1}}+\mathbf{1}}^{\mathbf{S_{max.2}}} \mathbf{N_i}, \tag{3.1}$$

where N is the number of states that need more solution steps; $S_{max.1}$ is the maximum solution steps of the first shape; $S_{max.2}$ is the maximum solution steps of the second shape; and N_i is the number of states in the depth i.

According to Equation (3.1), for all generated states, 88.35% of states could provide fewer solution steps in the 4 × 4 board than in the 2 × 8 for the 2×10^5 states. Furthermore, Equation (3.2) provides an increasing percentage of solution steps for different shapes.

$$\mathbf{S_{plus}} = \frac{|\mathbf{S_{max.1}} - \mathbf{S_{max.2}}|}{\mathbf{S_{max}}} \times \mathbf{100\%}, \tag{3.2}$$

where S_{plus} is the increasing percentage of solution steps; $S_{max.1}, S_{max.2}$ are the same as in Equation (3.1); and S_{max} is the total solution steps. Based on Equation (3.2), the results prove that the 4 × 4 board achieved 23.8% of steps better than the 2 × 8 board at 2×10^5 states. Overall, when increasing the number of states in both given boards, the 4 × 4 board performed better than the 2 × 8 board in terms of the number of steps. Next, we considered a 36-boxes size of the puzzle.

3.1.2 36-boxes Size

This size can sort 35 boxes. In this case, there are four different shapes (6 × 6, 4 × 9, 3 × 12, and 2 × 18). The same analysis as in the previous case with the size of 16-boxes was carried out. Figure 3.2 shows the solution steps of all states with the same number of boxes for different shapes.

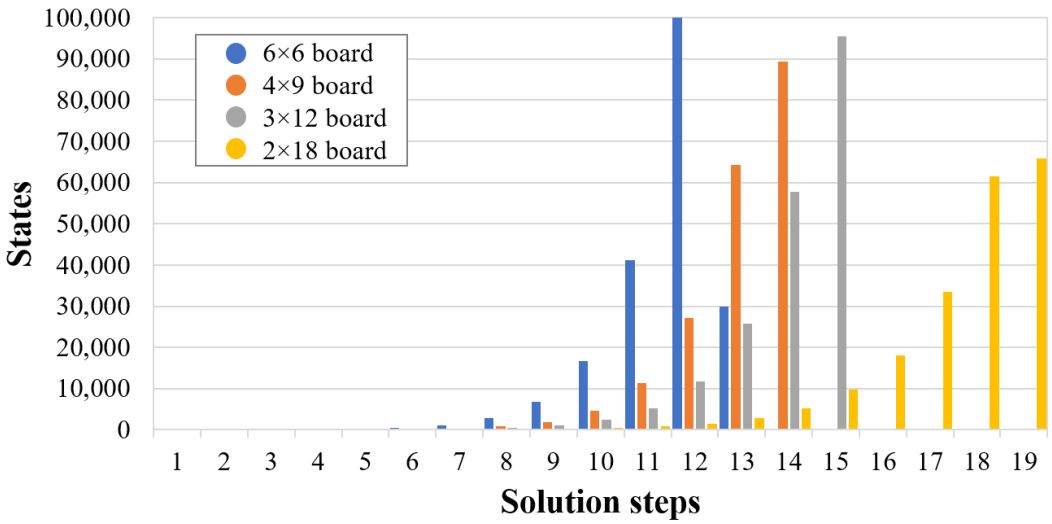


Figure 3.2. Comparison between the 6 × 6, 4 × 9, 3 × 12, and 2 × 18 board sizes for non-random states.

We observed the same trend in Figure 3.1 for our 16-boxes size of the puzzle in Figure 3.2. For all generated states, and referring to Equation (3.1), we confirmed that 44.63%, 76.60%, and 96.92% of states provided fewer solution steps in the 6 × 6 board than in the 4 × 9, 3 × 12, and 2 × 18 boards, respectively. Moreover, from Equation (3.2), the 6 × 6 board provided 7.14%, 13.33%, and 31.57% steps fewer than the 4 × 9, 3 × 12, and 2 × 18 boards, respectively. From these results, we deduced that the square shape of the puzzle had a better performance than the rectangular shape.

3.2 Board Size and Number

We discussed in Chapter 2, three strategies to carry out the sequencing process for more than 8 boxes. Since the first strategy which is increasing the size of the board had a limitation restricted by the puzzle capacity, we present and discuss in this chapter only the two other strategies.

3.2.1 Using Multi-Boards

In this section, we illustrate the results of two parameters, the time as a total number of steps, and the area used by the system.

3.2.1.1 Area for the Strategy of Using Multi-Boards

Based on Equation 2.6, we calculated the area used by the boards ignoring the main conveyor. Figure 3.3 illustrates the comparison between 8-puzzle, 15-puzzle, and 24-puzzle used for this strategy regarding the area.

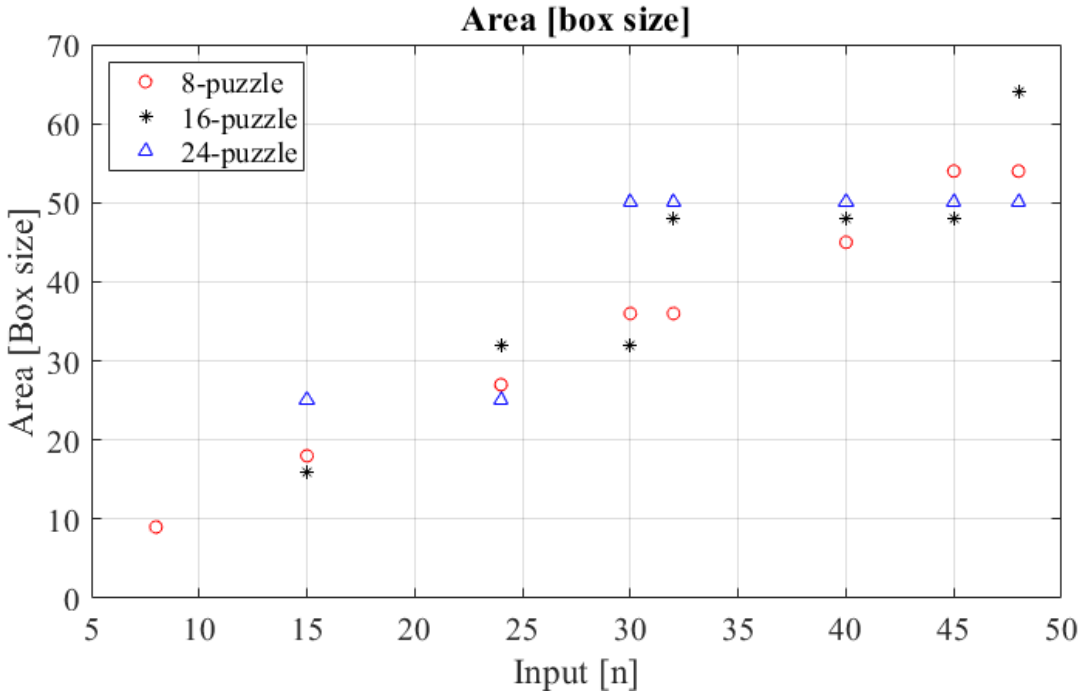


Figure 3.3. Comparison between 8-puzzle, 15-puzzle, and 24-puzzle used for multi-boards strategy regarding the area.

According to figure 3.3, we notice the variety in the area occupied by the system for different puzzle sizes, that is directly related to the deference number of used boards for the same input.

For example, for 32 boxes as an input, the number of boards used 8-puzzle board is 4 boards, using the 15-puzzle board is 3 boards, and using the 24-puzzle board is 2 boards. The areas of these three puzzles are 36 A_b , 48 A_b, and 50 A_b. In this example, it is clear that the 24-puzzle board is superior to the other boards.

In another example, for 48 boxes as an input, the areas are 54 A_b , 64 A_b, and 50 A_b using 8-puzzle, 15-puzzle, and 24-puzzle respectively.

3.2.1.2 Time for the Strategy of Using Multi-Boards

Based on Equation 2.8, Figure 3.4 illustrates the comparison between 8-puzzle, 15-puzzle, 24-puzzle used for this strategy regarding the total solution steps in the worst case which is considered the maximum solution steps of the puzzle.

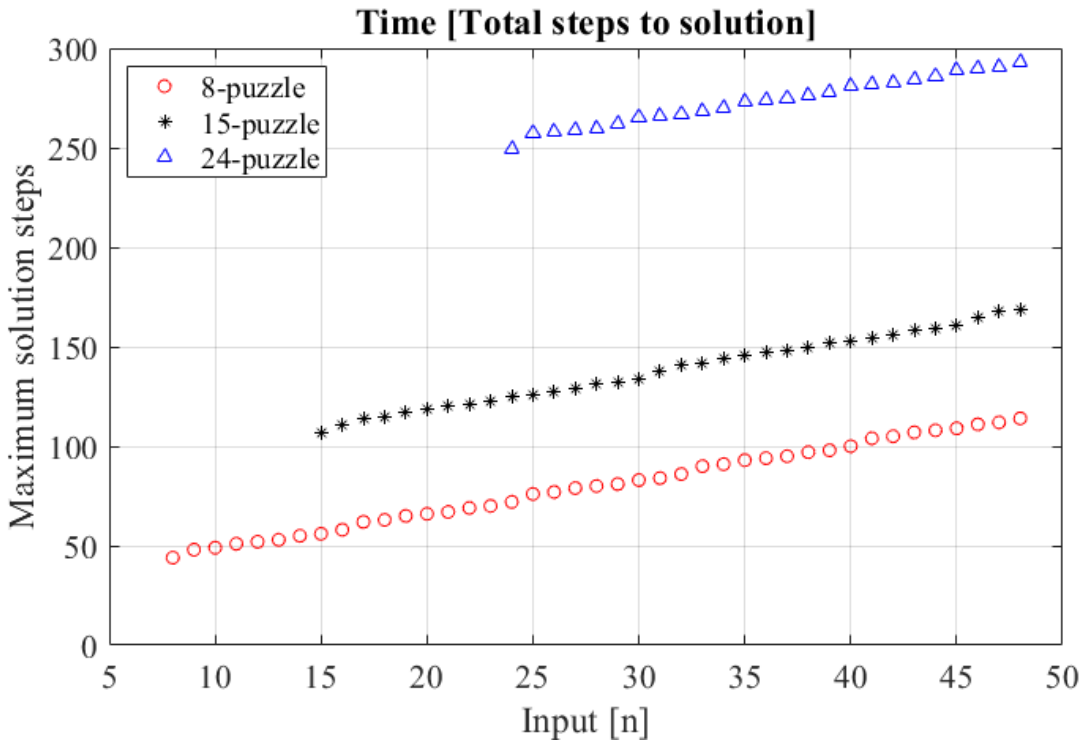


Figure 3.4. Comparison between 8-puzzle, 15-puzzle, and 24-puzzle used for multi-boards strategy regarding the total solution steps.

According to figure 3.4, we verified the superior of 8-puzzle board size to other boards sizes. The reason for that is the significant increase in the maximum solution steps for these puzzles as is illustrated in table 2.4.

Since the results vary regarding the area and time between different boards sizes, we need to find a compromise between the area and the time to evaluate the best board size for this strategy. Using the Selection Index theory may make us able to compromise between the area and the time for this strategy.

3.2.1.3 Selection Index Theory

As we needed to compromise between the area used by the system and the time of sequencing for this strategy, we used the Selection Index theory.

In animal and plants breeding, the breeding value is used by the definition of Estimation of Breeding Value (EBV), this estimation is calculated based on

individual observed phenotypes information, in addition to the information from relatives and correlated traits.

To combine information from different sources, the researchers used the Selection index and they expressed the EBV as an index, weighing different types of information. Equation 3.3 illustrates the classical Selection Index for a combination of different sourced information [44].

$$\boldsymbol{EBV = Index = b_1X_1 + b_2X_2 + \ldots + b_nX_n} \quad (3.3)$$

where EBV is the estimation of breeding value and b_1, b_1, …, b_n are the weights and X_1, X_2, …, X_n are phenotypic information sources.

In this study, we used the same theory to compromise between the area and time. Firstly, we normalized the results of the area and time to unify the range of the y axis. we rescaled the y axes of both time and area from 0 to 1. Then, we weighed equally the area and the time, and we keep to the logistical managers to evaluate the importance of these parameters. Equation 3.4 illustrates the use of the classical selection index for area and time.

$$\boldsymbol{I = +0.5\,A + \ 0.5\,S_{total}} \quad (3.4)$$

where $\boldsymbol{S_{total}}$ is the total solution steps which refer to the solution time; and A is the area used by the system.

Since we aimed to reduce the used area and minimize the solution time, the smaller the Index is the better strategy we get. Figure 3.5 illustrates the index of both area and time after normalizing (rescale) their results.

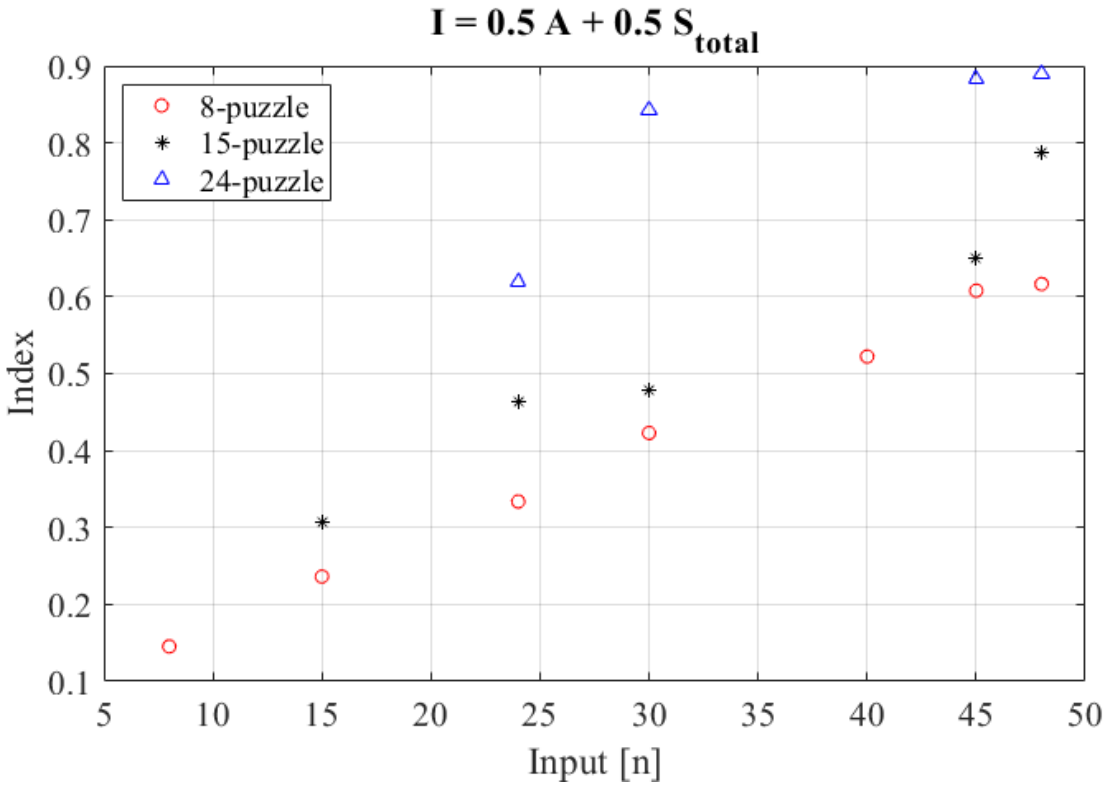


Figure 3.5. Selection index for normalized time and area of multi 8,15 and 24 boards for 48 input

According to figure 3.5, we observe the superior of using multi 8-puzzle size boards to other boards in this strategy.

3.2.2 Adding a Buffer Line

The same two parameters discussed to evaluate the strategy will be considered.

3.2.2.1 Area for the Strategy of Adding Buffer Line

We compared the area for this strategy with the area of using multi-8-puzzle boards illustrated in Section 3.2.1.1. Based on Equations 2.6, and 2.9, we get Figure 3.6 which illustrates the comparison between muti-8-puzzle boards and adding buffer line regarding the system area for 48 boxes as an input.

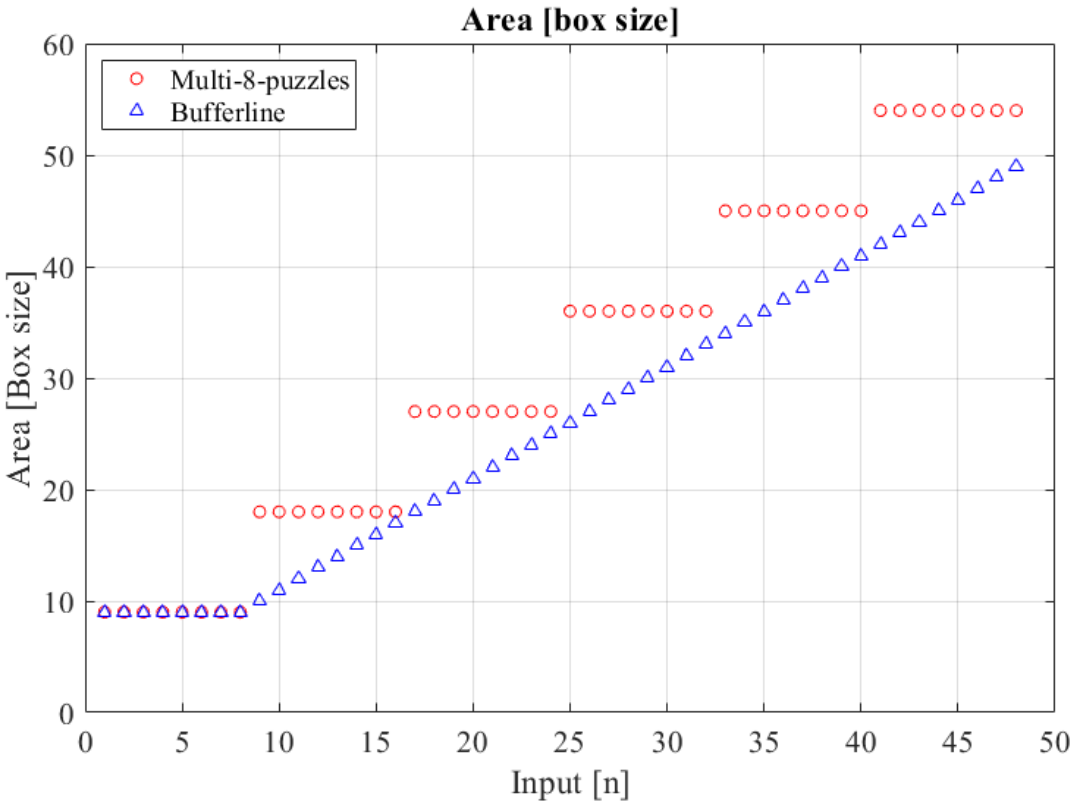


Figure 3.6. Comparison between muti-8-puzzle boards and adding buffer line regarding the system area.

In figure 3.6, we noticed that using a buffer line reduced the used area. The reason is the linearity of the area in this strategy, while in a multi-board strategy the area depended on the capacity of the puzzle which affects also the number of used boards.

3.2.2.2 Time for the Strategy of Adding Buffer Line

According to Equations 2.8, and 2.10, Figure 3.7 illustrates the comparison between muti-8-puzzle boards and adding buffer line regarding the total solution steps.

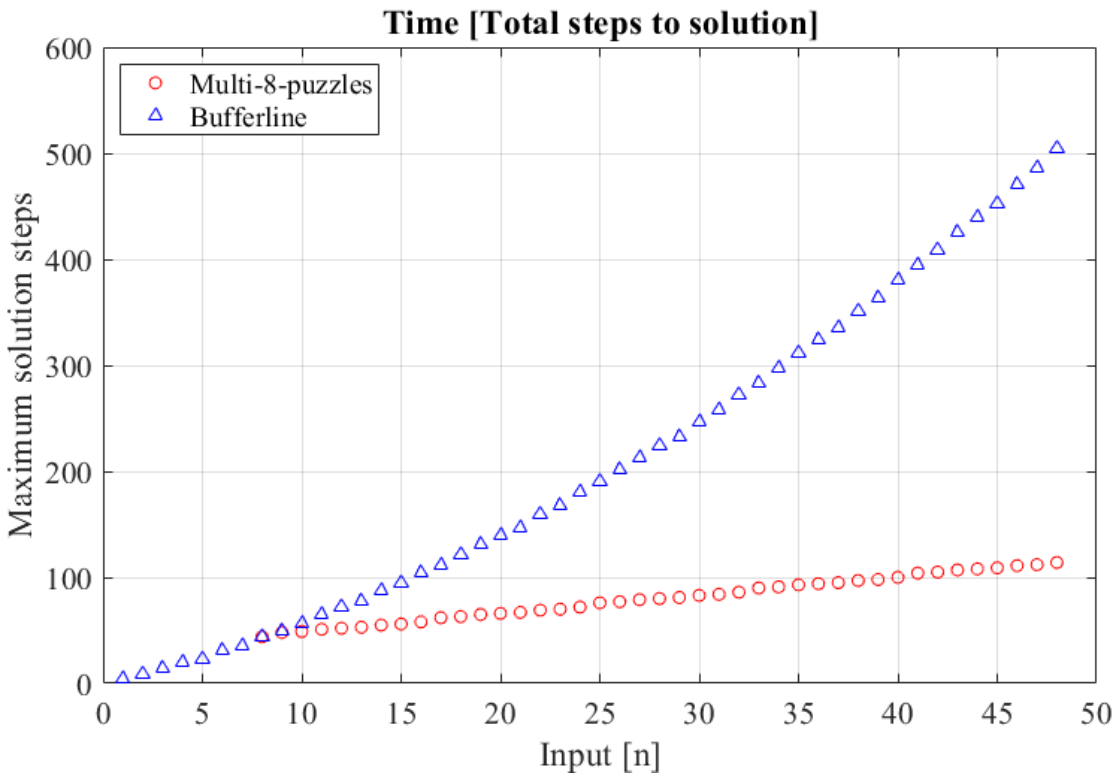


Figure 3.7. Comparison between muti-8-puzzle boards and adding buffer line regarding the total solution steps.

We noticed that adding a buffer line increases the total number of steps compared to the strategy of multi-boards with an 8-puzzle board size.

3.2.2.3 Index for the Strategy of Adding Buffer Line

As same as the strategy of using multi-boards, we used the selection index theory to compromise between the area and time for the strategy of adding buffer line.

For comparison and selection index, we compared the strategy of adding buffer line with the strategy of using multi-board with the size of 8-puzzle board.

Based on Equation 3.4 we calculated the index for these strategies. Figure 3.8 illustrates the index of both area and time in this strategy and multi-8-puzzle boards after normalization.

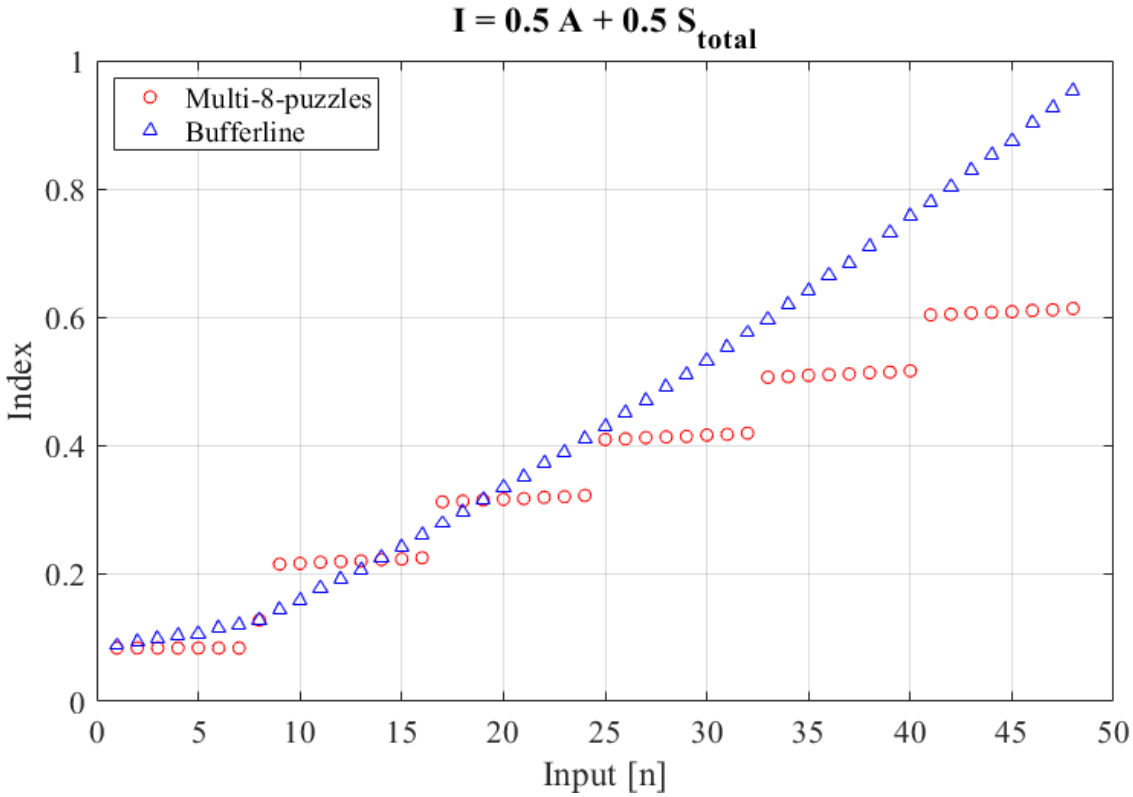


Figure 3.8. Selection index for normalized time and area of multi 8-puzzle boards and buffer line for 48 input

Based on figure 3.8, the strategy of using multi-boards with the 8-puzzle board size was better than the strategy of adding a buffer line to the system.

Figure 3.9 illustrates the curve fitting of the multi-boards strategy with three different board sizes and adding buffer line strategy. The curve fitting gives an estimation of the trend of indices for different input numbers.

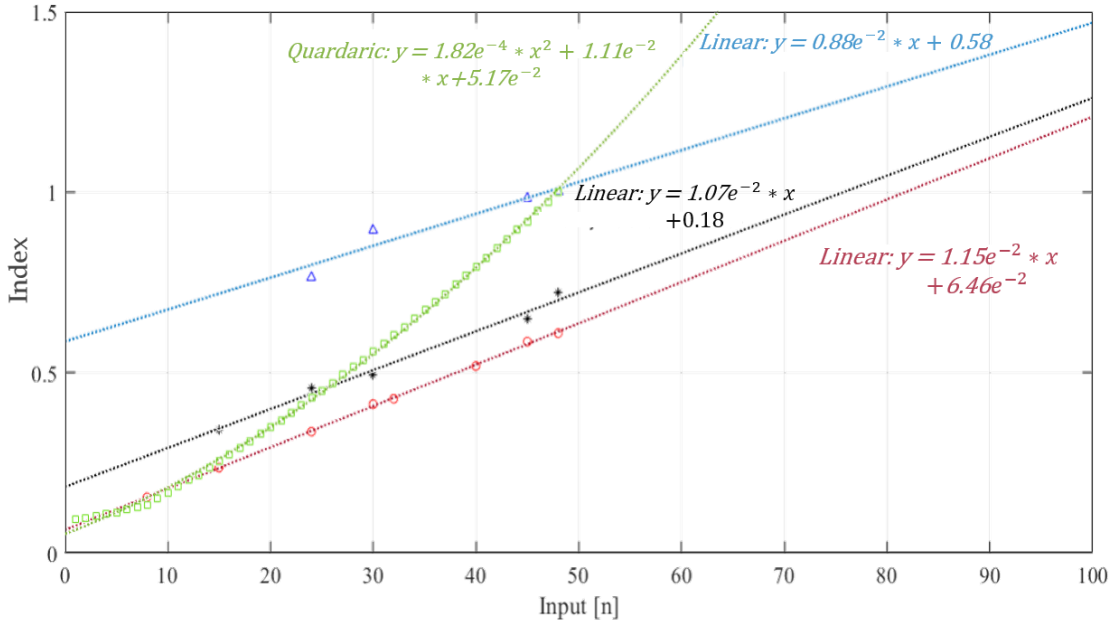


Figure 3.9. Curve fitting for multi-boards and adding buffer line strategies with three different board sizes.

From the figure, we can conclude the superiority of the strategy of using multi-boards with the 8-puzzle board size to other strategies and sizes.

3.3 Number of Blanks

To compromise between the number of blanks and the puzzle capacity, the same analyses of the strategy of using multi-boards with the size of 8-puzzle regarding the area and total solution steps were carried out. In this analysis, we considered the 8-puzzle with a different number of blanks. Figure 3.10 illustrates the area of the system for a different number of blanks in the strategy of using multi—puzzle boards.

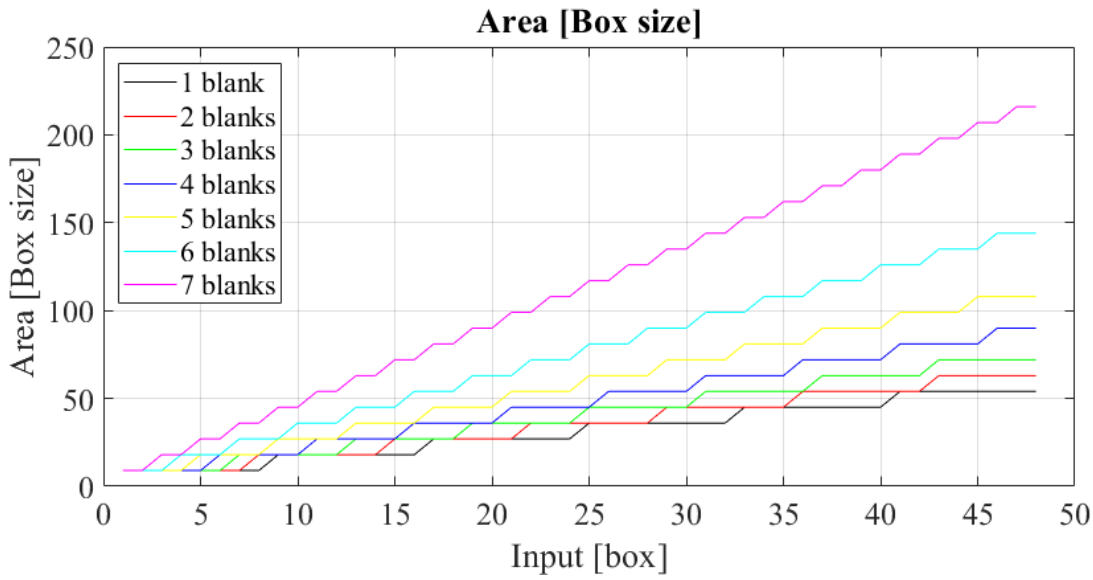


Figure 3.10. The area of multi 8-puzzle boards for different numbers of blanks.

As shown in the figure, increasing the number of blanks will always increase the system area. The reason is that increasing the blanks will decrease the puzzle capacity, and increase the number of boards as Equation 2.7. Figure 3.11 illustrates the total solution steps in the strategy of multi-8-puzzle boards for different numbers of blanks.

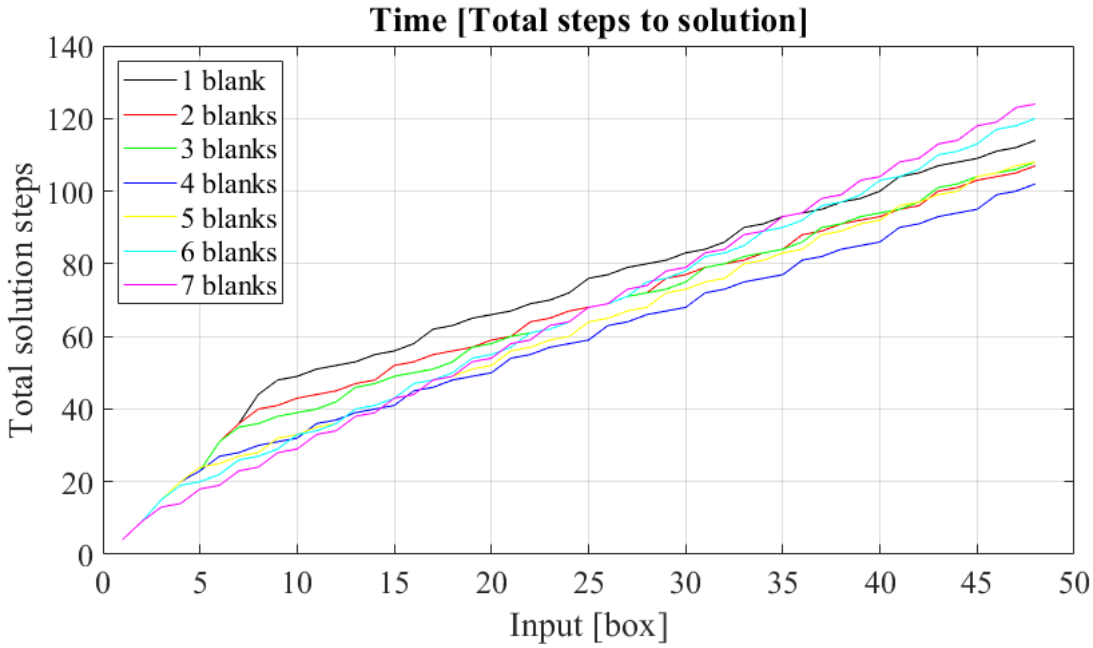


Figure 3.11. The total solution steps of multi 8-puzzle boards for different numbers of blanks.

According to Figure 3.11, 4 blanks provided the shortest solution steps.

The reason behind these results is the big difference in the number of boards which is the coefficient of the presorting steps as in Equation 2.8.

For example, for 30 boxes as an input of the system. 1 blank puzzle needed 4 boards while 4 blanks needed 6 boards. Even though the maximum solution steps is decreasing by increasing slightly the blank, the coefficient of this parameter is always 1.

Figure 3.12 illustrates the parameter of Equation 2.8 for 30 boxes regarding different numbers of blanks. As the total number of steps, 4 blanks provide the shortest solution steps compared to other puzzles with different numbers of blanks.

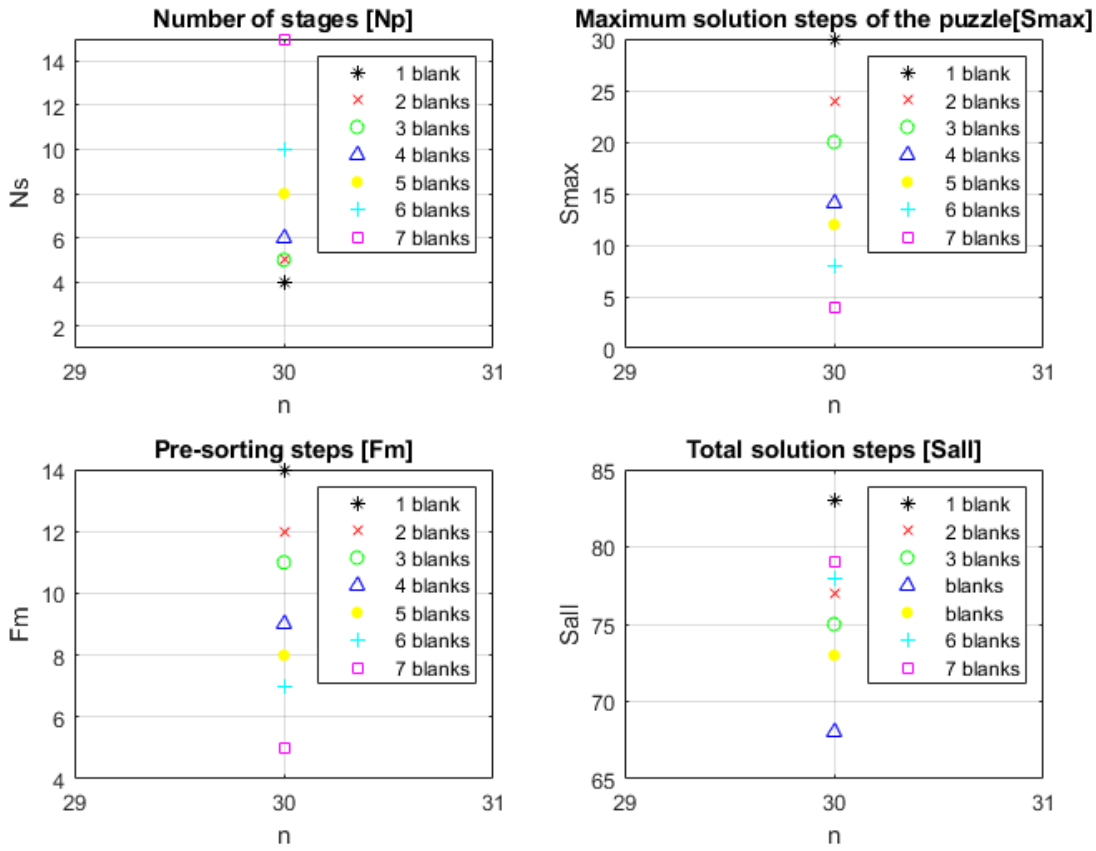


Figure 3.12. The parameter of Equation 5.5 for 30 boxes regarding different numbers of blanks.

Same analysis as for the strategies described before in this Chapter to evaluate the system regarding different numbers on blanks. We used the selection index theory to compromise between the area and time.

Figure 3.13 illustrates the index for multi-8-puzzle boards with different numbers of blanks.

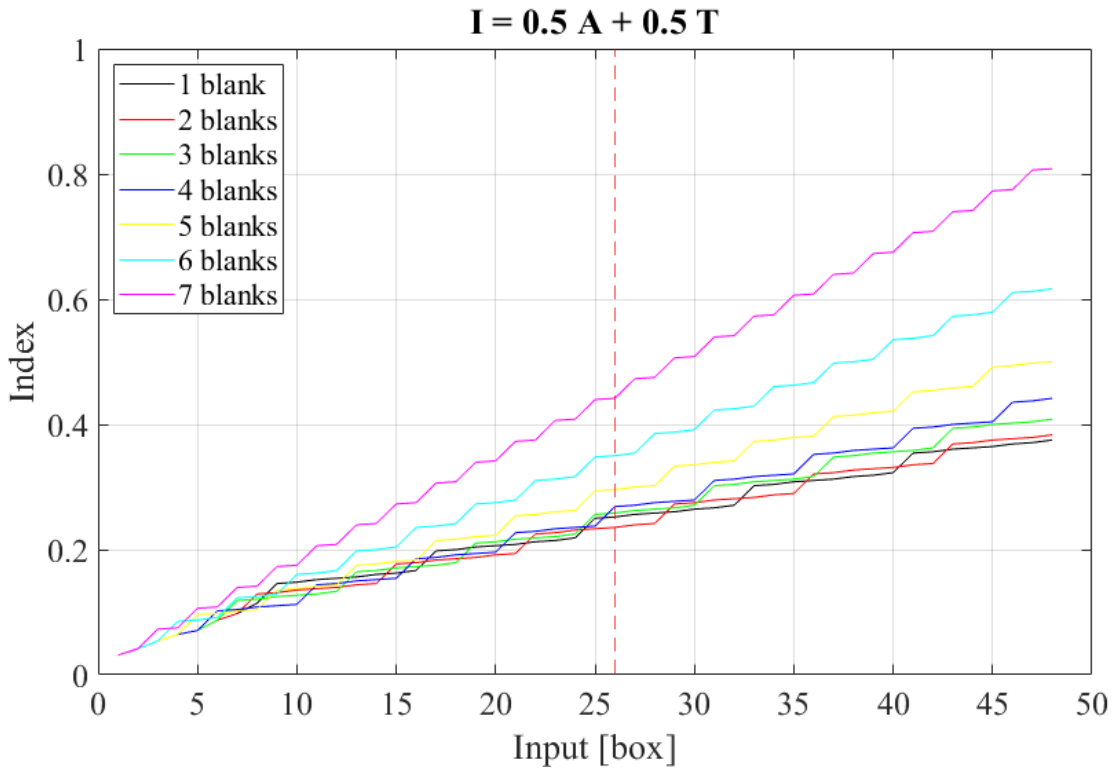


Figure 3.13. The index for multi-8-puzzle boards with different numbers of blanks.

According to figure 3.13, up to 25 boxes, the behavior of less than or equal to 4 blanks has almost the same index with very slight changes, while more than 4 blanks the puzzles have a bigger index.

For greater than 25 boxes, we noticed that 1 and 2 boxes have the almost same index, while 3 and 4 blanks started to give a bigger index than 2 and 1 blank. Until now, even though, these puzzles have more than one blank, we slide only one blank each step. Next, we investigated the double switching effect on the solution steps.

3.4 Double Switching

As described in Chapter 2, we carried out double switching in one step. We have two different cases: the general case where the blanks are placed randomly in the puzzle and the case where the blanks are placed in the corner

of the puzzle. We present the effect of double switching on the solution steps for both cases.

3.4.1 General Case

We investigated the new concept to solve 8-puzzle with 2 blanks for all configurations in the general case where the blanks are randomly placed in the puzzle.

Figure 3.14 illustrates the number of possible simultaneous double switching for an 8-puzzle with 2 blanks.

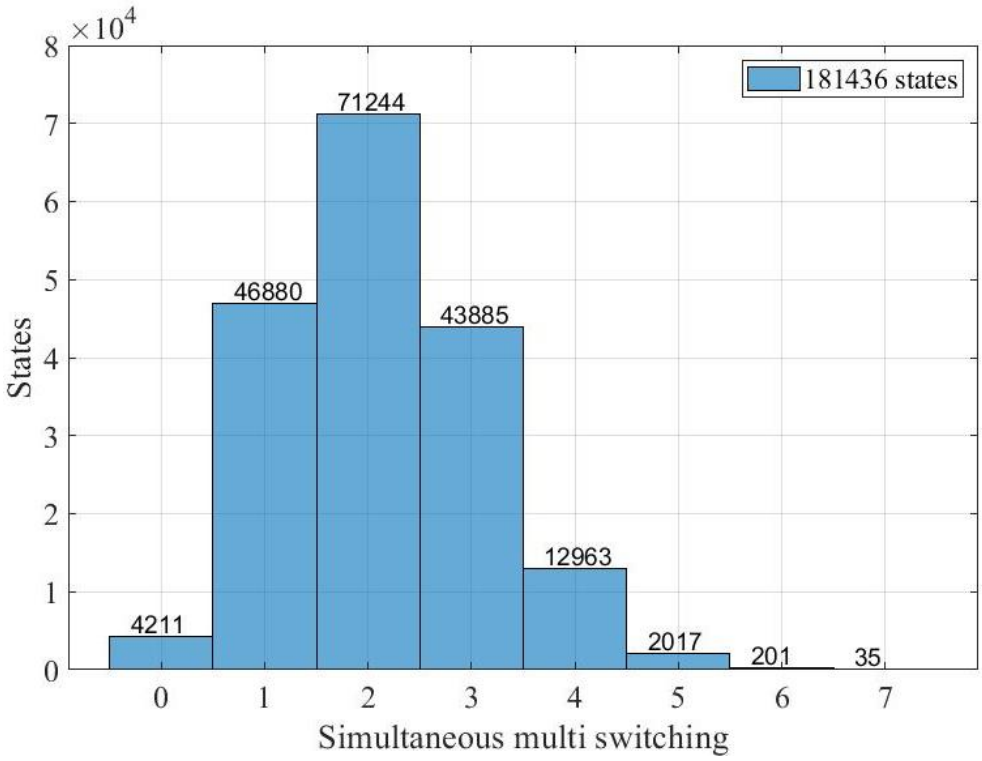


Figure 3.14. The number of simultaneous double switching for 8-puzzle with 2 blanks.

In figure 3.14, we noticed that the maximum states are 181436 states, because, the first 4 states in the tree take less than 2 steps to reach the goal state.

According to the figure and the analysis steps described, we reduced the number of solution steps for 177225 states out of 181436 states by a minimum of 1 step, an average of 2 steps, and a maximum of 7 steps.

To evaluate the reduction percentage by carrying out double switching in one step, we considered the average solution steps of the puzzle with 2 blanks as illustrated in Table 2.6. In addition, we considered the average reduction steps which are 2 steps.

Figure 3.15 illustrates the reduction percentage of solution steps for 8-puzzle with 2 blanks.

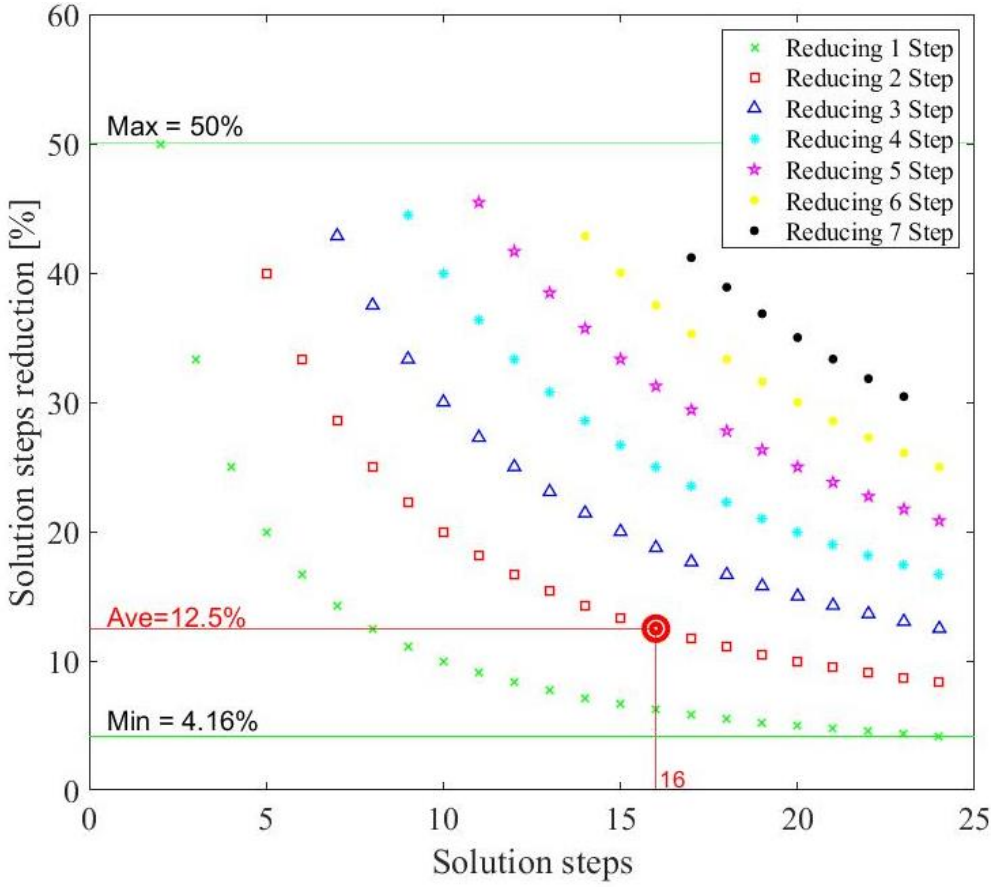


Figure 3.15. Reduction percentage of solution steps for 8-puzzle with 2 blanks.

According to the figure, by considering the average solution steps and average reduction steps, we got a 12.5% reduction percentage of the solution steps.

The results shown in figure 3.15 are summarized in table 3.3.

Table 3.3. The reduction percentage of solution steps for 8-puzzle with 2 blanks.

Double-switching	Minimum	Maximum	Average (16 solution steps)
1 time (Reduce 1 step)	4.16%	50%	6.25%
2 times (Reduce 2 steps)	8.33%	40%	12.5%
3 times (Reduce 3 steps)	12.5%	42.85%	16.66%
4 times (Reduce 4 steps)	16.66%	44.44%	22.22%
5 times (Reduce 5 steps)	20.83%	45.45%	26.31%
6 times (Reduce 6 steps)	25%	42.85%	31.57%
7 times (Reduce 7 steps)	30.43%	41.17%	33.33%

According to table 3.3, the minimum percentage of steps reduction is 4.16% and the maximum is 50%.

3.4.2 Blanks Placed in the Corner of the Puzzle

We assumed that practical implementation in a real-world warehouse requires placing the blanks in the corner of the puzzle.

The same analyses for double switching in the general case were conducted considering our assumption for practical implementation requirement.

Figure 3.16 illustrates the number of possible simultaneous double switching for an 8-puzzle with 2 blanks placed in the corner of the puzzle.

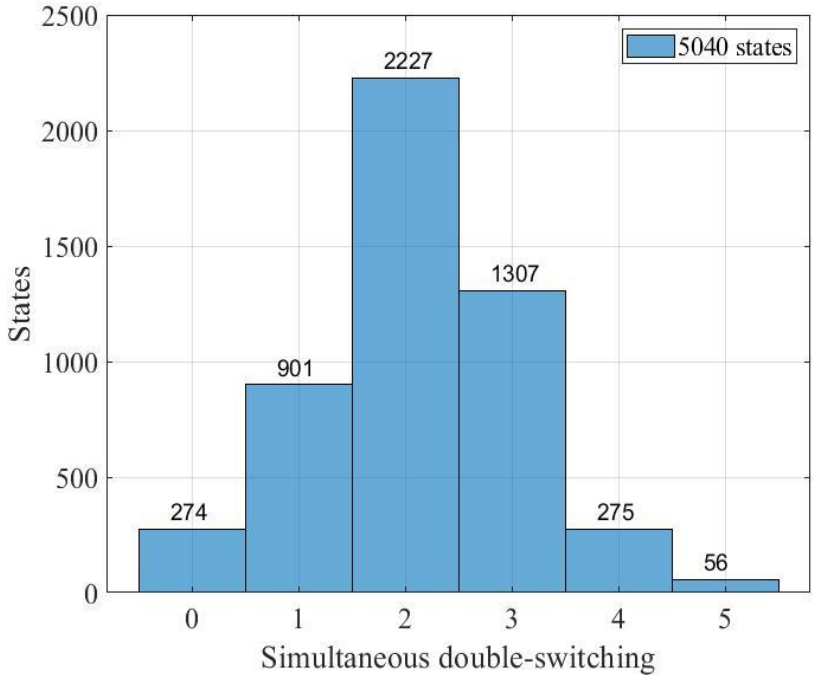


Figure 3.16. The number of simultaneous double switching for 8-puzzle with 2 blanks placed in the corner of the puzzle.

As shown in the figure, the total states, in this case, are 5040 states. We reduced the number of solution steps for 4766 states out of 5040 states by a minimum of 1 step, an average of 2 steps, and a maximum of 5 steps.

Considering the case of placing the planks in the puzzle corner, we generated all the states and calculate the maximum and average solution steps for an arbitrary number of blanks for 8-puzzle as illustrated in table 3.4.

Table 3.4. The effect of an arbitrary number of blanks on the solution steps for an 8-puzzle with 2 blanks placed in the corner of the puzzle.

Number of blanks	Maximum capacity	Maximum states	Maximum solution steps	Average solution steps
1	8	20160	30	22.14
2	7	5040	24	16.27
3	6	720	20	13.17
4	5	120	14	10
5	4	24	12	7.39
6	3	6	8	5.6
7	2	2	4	4

According to table 3.4, the average solution steps are 16 steps. And as shown in figure 3.16, the average reduction steps are 2 steps. Keep this in mind, we analyzed the reduction percentage of solution steps when carrying out double switching in one step.

Figure 3.17 illustrates the reduction percentage of solution steps for an 8-puzzle with 2 blanks placed in the corner of the puzzle.

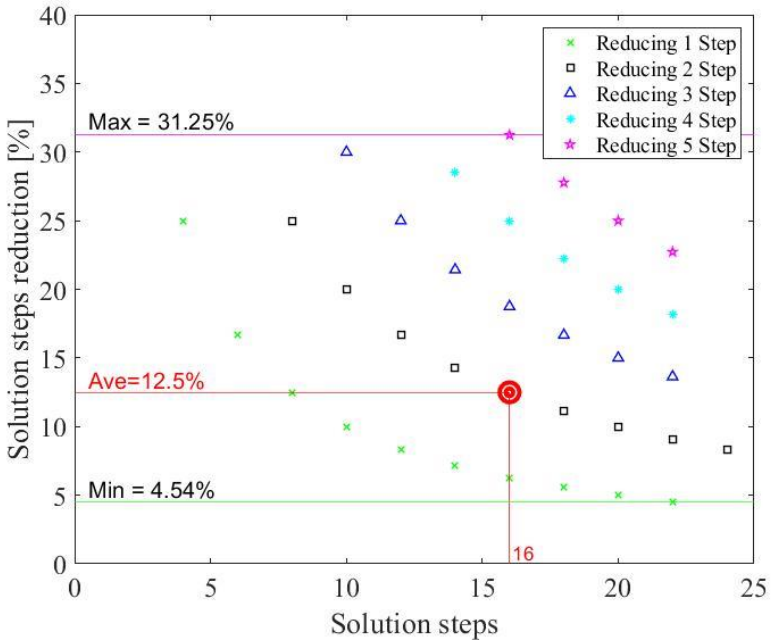


Figure 3.17. Reduction percentage of solution steps for an 8-puzzle with 2 blanks for practical implementation requirement.

The summary of the results shown in the figure is illustrated in table 3.5.

Table 3.5. The reduction percentage of solution steps for 8-puzzle with 2 blanks placed in the corner of the puzzle.

Double-switching	**Minimum**	**Maximum**	**Average (16 solution steps)**
1 time (Reduce 1 step)	4.54%	25%	6.25%
2 times (Reduce 2 steps)	8.33%	25%	12.5%
3 times (Reduce 3 steps)	13.63%	30%	16.66%
4 times (Reduce 4 steps)	18.18%	28.57%	22.22%
5 times (Reduce 5 steps)	22.72%	31.25%	27.77%

According to table 3.5, the minimum percentage of steps reduction is 4.54%, the maximum is 31.25%, and the average is 12.5%.

To evaluate the improvement of applying the block movement on the total solution steps in the system, the same analyses as in 3.4.1 and 3.4.2 are carried out here.

Figures 3.18 and 3.19 illustrate the number of simultaneous double switching for 8-puzzle with 2 blanks after applying the block movement in both general case and the case of placing the blanks in the corner of the puzzle, respectively.

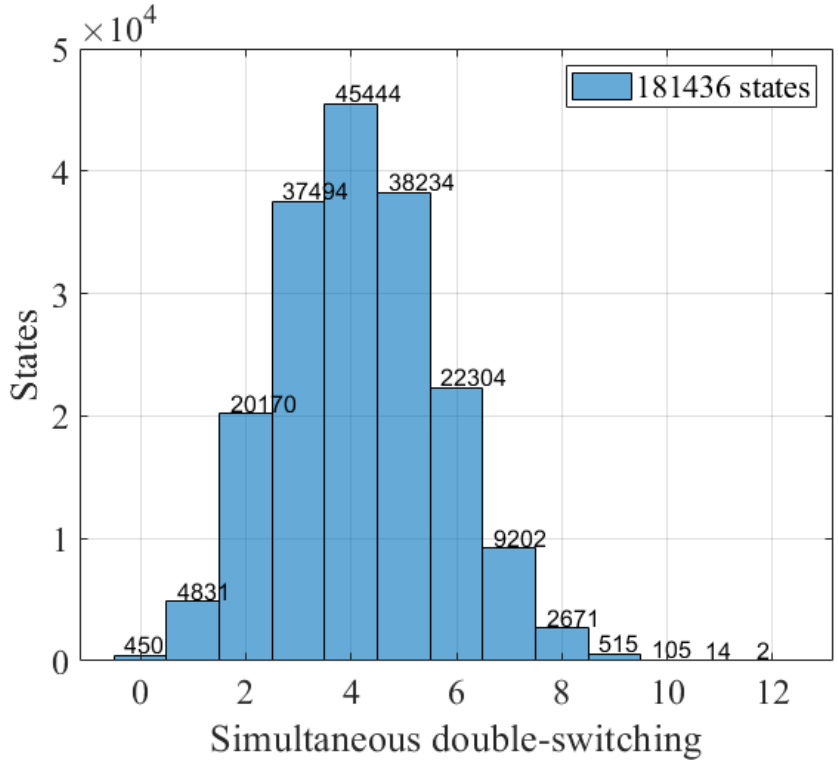


Figure 3.18. The number of simultaneous double switching with block movement for 8-puzzle with 2 blanks.

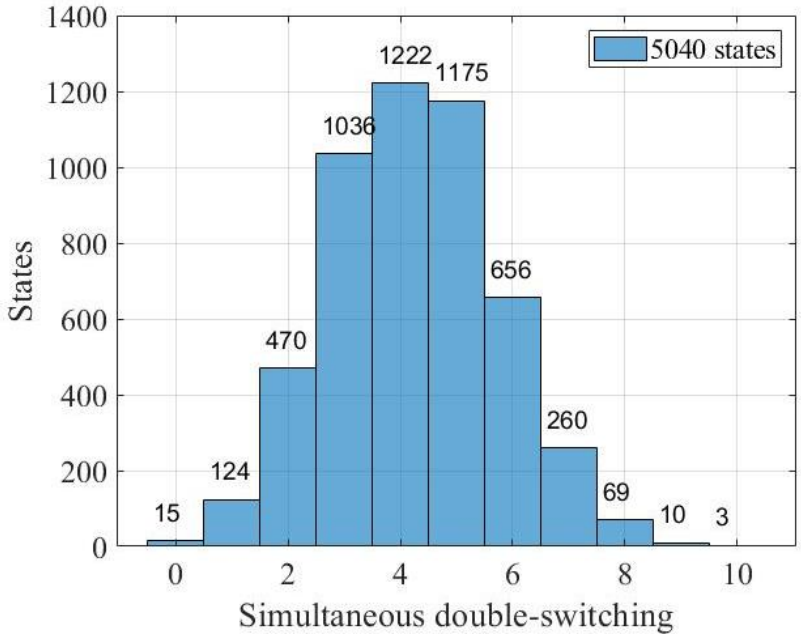


Figure 3.19. The number of simultaneous double switching with block movement for 8-puzzle with 2 blanks placed in the corner of the puzzle.

According to figure 3.18, We reduced the number of solution steps for 180,986 states out of 181,436 states by a minimum of 1 step, an average of 4 steps, and a maximum of 12 steps.

In figure 3.19, We reduced the number of solution steps for 5025 states out of 5040 states by a minimum of 1 step, an average of 4 steps, and a maximum of 10 steps.

For getting the reduction percentage of steps after applying the block movement, we concede the average solution steps of the 8-puzzle with 2 blanks (16 steps) and the average reduction steps (4 steps). Figures 3.20 and 3.21 illustrate the reduction percentage in the general case and the practical implementation requirement, respectively.

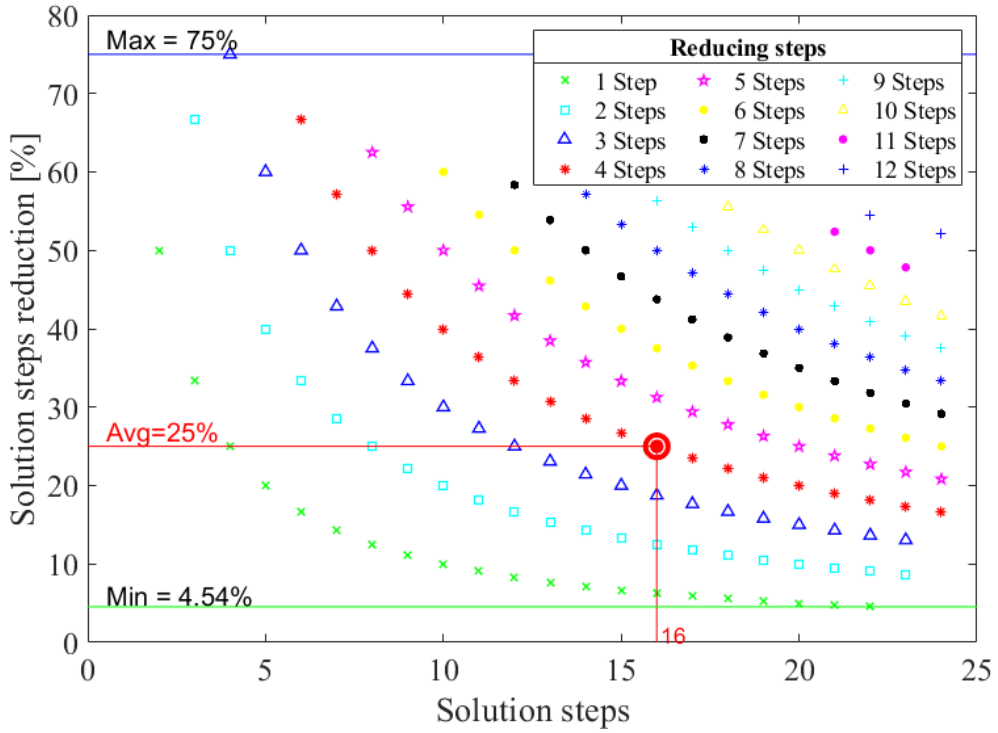


Figure 3.20. Reduction percentage of solution steps after applying block movement for 8-puzzle with 2 blanks.

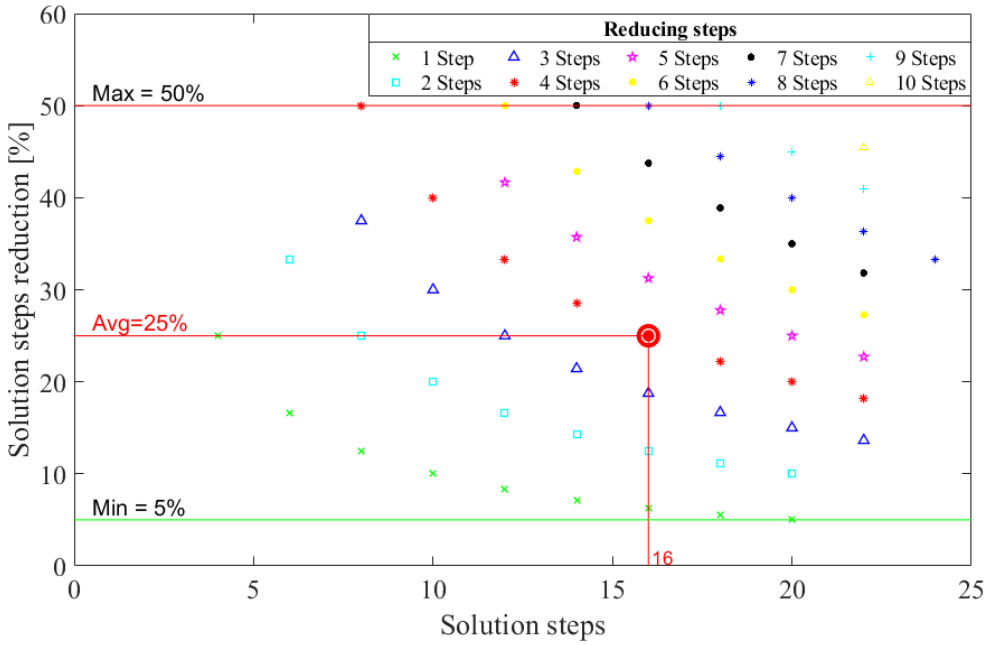


Figure 3.21. Reduction percentage of solution steps after applying block movement for 8-puzzle with 2 blanks placed in the corner of the puzzle.

The summary of the results shown in figures 3.20 and 3.21 are illustrated in tables 3.6 and 3.7, respectively.

Table 3.6. The reduction percentage of solution after applying block movement steps for 8-puzzle with 2 blanks.

Double-switching	Minimum	Maximum	Average (16 solution steps)
1 time (Reduce 1 step)	4.54%	50%	6.25%
2 times (Reduce 2 steps)	8.69%	66.66%	12.5%
3 times (Reduce 3 steps)	13.4%	75%	18.75%
4 times (Reduce 4 steps)	16.66%	66.66%	25%
5 times (Reduce 5 steps)	20.83%	62.5%	31.25%
6 time (Reduce 6 steps)	25%	60%	37.5%
7 times (Reduce 7 steps)	29.16%	58.33%	43.75%
8 times (Reduce 8 steps)	33.33%	57.14%	50%
9 times (Reduce 9 steps)	37.5%	56.25%	56.25%
10 times (Reduce 10 steps)	41.66%	55.55%	-
11 times (Reduce 11 steps)	47.82%	52.38%	-
12 times (Reduce 12 steps)	52.17%	54.54%	-

Table 3.7. The reduction percentage of solution steps after applying block movement for 8-puzzle with 2 blanks placed in the corner of the puzzle.

Double-switching	Minimum	Maximum	Average (16 solution steps)
1 time (Reduce 1 step)	5%	25%	6.25%
2 times (Reduce 2 steps)	10%	33.33%	12.5%
3 times (Reduce 3 steps)	13.63%%	37.5%	21.42%
4 times (Reduce 4 steps)	18.18	50%	25%
5 times (Reduce 5 steps)	22.72%	41.66%	31.25%
6 time (Reduce 6 steps)	27.27%	50%	37.5%
7 times (Reduce 7 steps)	31.81%	50%	43.75%
8 times (Reduce 8 steps)	33.33%	50%	50%
9 times (Reduce 9 steps)	40.9%	50%	-
10 times (Reduce 10 steps)	-	45.45%	-

According to table 3.6, the minimum percentage of steps reduction is 5%, the maximum is 50%, and the average is 25%.

According to table 3.7, the minimum percentage of steps reduction is 4.54%, the maximum is 75%, and the average is 25%.

From tables 3.6 and 3.7, we confirmed the improvement in the system regarding the solution steps after applying the block movement.

3.5 Managerial Impact

To evaluate the proposed system and explore the impact of implementing the puzzle-based concept in products sequencing, we compared the proposed system with the other used systems. In addition, we compared the puzzle system with the traditional sorting algorithm (Dual-Pivot Quicksort algorithm).

Firstly, the puzzle-based sequencing system was compared with the GridSequence system developed by Gue et al. [11] with respect to the floor used area and sequencing time. In order to evaluate the utilization of floor space in the sequencing system, we calculated the area used by the puzzle-based system based on Equation 2.6 (we considered the strategy of using multi-boards with the size of 8-puzzle). For the GridSequence system, we considering 1 additional column and one additional row to the grid. Thus, the area is calculated as $(n + 1) \times (m + 1)$, where $(i = n * m)$, where i is the number of boxes that need to be sequenced.

Table 3.8 illustrates the used area in the puzzle-based system versus the GridSequence system for sequencing different numbers of boxes.

Table 3.8. The used area in a puzzle-based system vs. a GridSequence system.

Number of boxes (i)	Puzzle-Based System $\text{Area} = [(C_s + 1)N_s] \times A_b$		GridSequence System Area = (n + 1) × (m + 1) × A_b	
8	C_s = 8, N_s = 1	9	n = 2, m = 4	15
32	C_s = 8, N_s = 4	36	n = 5, m = 7	42
48	C_s = 8, N_s = 6	54	n = 6, m = 8	63
96	C_s = 8, N_s = 12	108	n = 8, m = 12	117

As shown in Table 3.8, the puzzle-based system can provide a less used area than the GridSequence system. Better space utilization is quantified, with a practical example; to sequence 32 boxes with sizes of 35 cm × 35 cm = 0.1225 m2, GridSequence would occupy 5.14 m2, while the proposed puzzle-based would occupy 4.41 m2. Therefore, a puzzle-based sequencing system is recommended to reduce the space as well as reduce the cost.

Second, we compared the proposed system with the GridSequence system regarding the sequencing time. In the puzzle-based system, we considered the multi-boards with the size of 8-puzzle, and we assumed that one step is carried out in 1 second. In addition, we add 3 steps to empty each board (assuming that every 3 boxes will be out of the puzzle simultaneously as on block). In the GridSequence we took into consideration 1 and 2 columns, and we considered the case of aspect ratio equals to 1 to match our board aspect ratio. Table 3.9 illustrates the sequencing time in the puzzle-based system versus the GridSequence system for sequencing 96 boxes.

Table 3.9. The sequencing time in a puzzle-based system vs. a GridSequence system for 96 boxes.

Number of boxes (i)	Puzzle-Based System Equation 2.8		GridSequence System Gue et al. [11]	
96	N_s = 12	234 Sec.	No. of Column=1	≈320 Sec.
96	N_s = 12	234 Sec.	No. of Column=2	≈313 Sec.

As shown in the table, the puzzle-based system can sequence 96 boxes in 234 seconds (Recall that 1 step is carried out in 1 second, which is realistic time in real-word). As a result, the proposed system significantly reduces the sequencing time compared with the GridSequence system.

Further comparison is carried out with traditional sorting algorithms such as the Dual-Pivot Quicksort algorithm. In the Dual-Pivot Quicksort algorithm, we chose two pivots and the algorithm can be described as follows:

1. Define the first and the last elements in the series as pivot 1 (P1) and pivot 2 (P2) respectively, and the remaining elements are divided into three parts: in part I, the elements that are smaller than P1, in part III, the elements that are bigger than P2. the rest of elements are placed in part II as illustrated in figure 3.22.

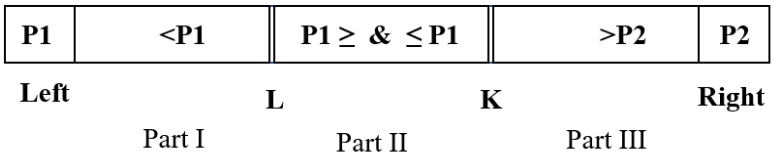
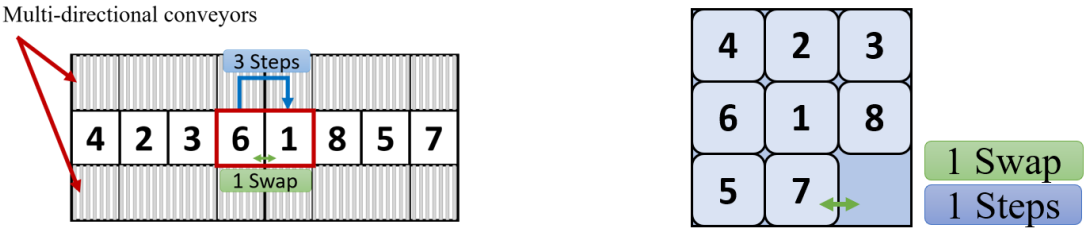


Figure 3.22. Concept of Dual-Pivot Quicksort algorithm.

2. Swap P1 with the last element of part I, and swap P2 with the first element of part III.
3. Repeat steps 1 and 2 for Parts I, II, III.

   The average number of swaps of the Quicksort algorithm with 2 pivots is (0.8*n* ln(n)) [45].

Since the Dual-Pivot Quicksort algorithm has a smaller number of swaps than classical Quicksort, we compared a concept of implementing this algorithm to sort 8 items using flexible multi-directional conveyors with puzzle sorting concept as illustrated in figure 3.23.



Quick sort with multi-directional conveyors

Puzzle sorting concept

Figure 3.23. Principle of swap & step in Quicksort algorithm with multi-directional conveyors system and puzzle sorting concept.

Table 3.10 illustrates the comparison between the puzzle concept and Dual-Pivot Quicksort algorithm regarding the area, and number of steps

Table 3.10. Comparison between the puzzle concept and Dual-Pivot Quicksort algorithm.

	Dual-Pivot Quicksort	n=8	Puzzle-based system	n=8
Area [box size]	3n	24	n+1	9
No. of steps [45]	Avg.= 0.8 × n × ln(n) × 3*	39.9	Max.= 31 (for n=8)	31

* Swapping two boxes needs at least 3 steps

According to the table, the puzzle provided fewer steps than the Quicksort algorithm, also the area used by the puzzle is less than that used by the flexible multi-directional conveyors.

We investigated the implementation of Quicksort algorithm utilizing puzzle movement concept for an example of 8 elements by applying the following steps:

1. Arrange the list on a 3×3 grid (3×3 puzzle grid).
2. Do partitioning considering the pivots (sub-targets), with putting in consideration that swapping two tiles should not change others sequence configuration.
3. Put the pivots in their proper positions (sub-target).

4. Repeat the algorithm for the unsorted partitions.

Figure 3.24 illustrates an example of implementing the Dual-Pivot Quicksort algorithm utilizing the sliding puzzle movement concept for 8 elements (assuming we can do multiple swapping simultaneously).

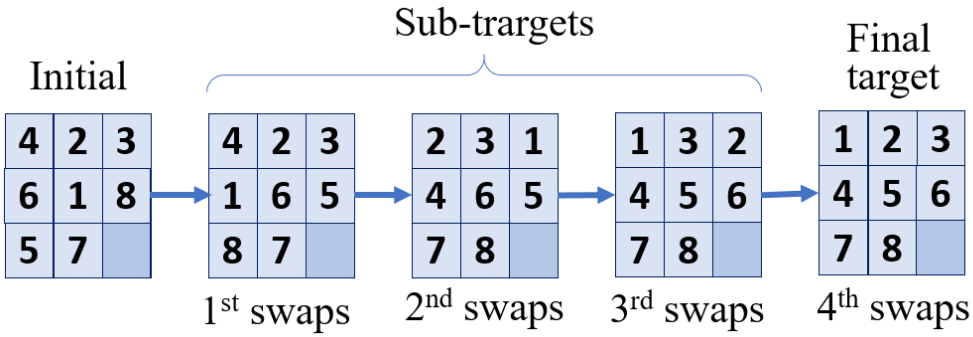


Figure 3.24. Implementation of Dual-Pivot Quicksort algorithm with 2 pivots utilizing sliding puzzle concept.

According to figure 3.24, swapping two elements needs several steps to move from one state to the next sub-target, and each sub-target is considered as a puzzle. Based on the previous example; in order to sort the list of 8 numbers, we had to solve the puzzle four times.

Table 3.11 illustrates the comparison between the puzzle concept and Dual-Pivot Quicksort algorithm used puzzle-based board regarding the area, and number of steps

Table 3.11. Comparison between the puzzle concept and Dual-Pivot Quicksort algorithm used puzzle-based board.

	Dual-Pivot Quicksort	n=8	Puzzle-based system	n=8
Area [box size]	3+1	9	n+1	9
No. of steps [45]	Avg.= 0.8 × n × ln(n) × 16*	212.9	Max.= 31 (for n=8)	31

* Average solution steps for 8-puzzle is 16

According to the table, the puzzle provided much less sequencing time than Dual-Pivot Quicksort.

Figure 3.25 illustrates the performance of the proposed method compared with other sequencing systems regarding the used area and the sequencing time.

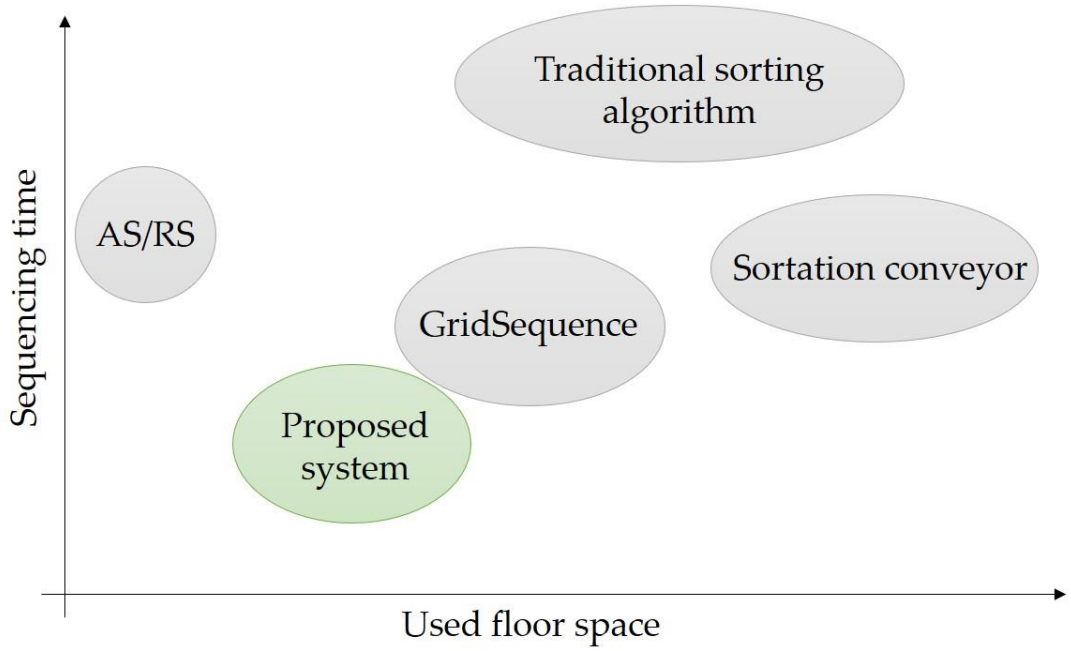


Figure 3.25 Performance comparison between the proposed method and other sequencing systems regarding the used area and the sequencing time.

According to the figure, the proposed puzzle-based sequence system provided better performance as a counterbalance between used floor space and the sequencing time.

3.6 Summary

In this chapter, we investigated the effect of the board shapes on the system regarding solution steps. We compared the same size of the puzzle with different shapes. The results showed that a square shape provided better performance than a rectangular one.

The results of the practical implementation strategies were presented. we concluded that the strategy of using multi-board with the 8-puzzle board size was the most suitable strategy regarding both the used system area and total solution steps.

We dealt in this Chapter with the puzzle with an arbitrary number of blanks. In the strategy of using multi-boards with an 8-puzzle board size, we carried out the sorting for different numbers of blanks for 8 to 48 boxes as an input of the system. As a result, if the number of input boxes is up to 25 boxes, increasing the blanks up to 4 blanks has a very slight effect, however, if the number of inputs grows above 25 boxes, 1 or 2 blanks shows almost the same behaviour regarding the area used by the system and the total solution steps. while increasing the blanks more than 2 gave an opposite effect on the system.

We investigated the effect of simultaneous double switching in one step on the system regarding maximum solution steps. Simultaneous double switching allows reducing the maximum solution steps by an average of 2 steps and as of 12.5% steps reduction percentage. Afterward, we improve the reduction percentage by applying the block movement concept. As a result, we reduced the solution steps by an average of 4 steps and an average of 25% steps as a reduction percentage.

The proposed system provided a higher floor space utilization and lower sequencing time compared with some systems and sorting algorithms.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

Item sequencing has become necessary in order to increase the efficiency of logistics operations. In this Dissertation, we focused on the material handling devices that could carry out the sequencing task. We developed a puzzle-based sequencing system with highly efficient floor space utilization as well as lower sequencing time. Different searching techniques were discussed, and the A-star algorithm was chosen to find the shortest solution for the puzzle. Furthermore, a pre-sorting process was proposed to overcome unsolvable configurations. In the pre-sorting process, we switched the last two items; therefore, different filling-in processes might affect the overall steps to reach the final goal of the puzzle.

Two shapes of the puzzle with the same size were considered to achieve the minimum number of solution steps. The results clarified a different number of states in the same level of the generated tree for both shapes with different sizes. For different puzzles, if we give a random state, there is a high probability that it will be in the tree with the higher number of states at the same level. Several factors were discussed with their effects on the puzzle solution steps. Based on the results of the numerical calculations, it can be concluded that a square shape can provide a shorter solution than a rectangular shape.

Practically, the 8-puzzle sequencing system is restricted by the puzzle capacity. Therefore, we proposed and discussed three strategies to meet the practical implementation in real-world warehouses where the need of sequencing a list of more than 8 items.

Our proposed strategies were:

1. Increasing the board size using different puzzle board sizes: In this strategy, we were still limited to the used puzzle capacity.
2. Using multi-boards: in this strategy, we used several boards placed along with the main conveyor, on these boards the sequencing processes were carried out in parallel. we compared 8,16 and 24-puzzle, and we observe that 8-puzzle board size performs better than other boards regarding the area used by the system and the total solution steps.
3. Adding buffer line: in this strategy, we added a buffer conveyor along with the main input conveyor. For the input boxes ordered more than 8, they will temporarily be buffered and resequencing in the next sequencing process. We compared the strategy of using multi-boards with 8-puzzle board size with the strategy of adding buffer line, we observe the superiority of using multi-8-puzzle boards.

Finally, we investigated the effect of increasing the blanks in the puzzle which reduced the maximum solution steps. Carrying out simultaneous double switching allowed us to reduce the maximum solution steps by an average of 2 steps which is a 12.5% steps reduction percentage. After the improvement by applying the concept of block movement, we were able to reduce the solution steps by an average of 4 steps which is a 25% steps reduction percentage. The best strategy for more than 8 boxes is using multi-boards along with the main feeding conveyor with the shape and size of 8-puzzle with 2 blanks.

Compared with other sequencing systems the proposed puzzle-based system provided a lower used area and highly efficient floor space utilization. Furthermore, the puzzle system achieved better performance regarding the sequencing time. These points are important parameters when considering designing a material handling device for products sequencing to reduce the capital, operational and variable costs including minimizing the cost of workforces.

4.2 Limitations and Future Work

4.2.1 Limitations

The limitation of the presented system can be represented from the Mechanical point of view. In the case where a conveyor module has a problem, the cell of this module will be considered as a broken cell or idle cell. Here we have 3 cases:

- The idle cell is in the puzzle's corner: we still be able to use the same algorithm, however, we can be able to sequence up to 7 boxes per board.
- The idle cell is along the edge of the puzzle: the maximum number of boxes that can be sequenced is 6 boxes, and we use the same algorithm.
- The idle cell is in the middle of the puzzle: here the sequencing process won't proceed anymore, thus we need to maintain the board.

This problem can be more considerable when more than one cell is broken in the board.

4.2.1 Future Work

As presented in this thesis, we used a pre-sorting process to overcome the unsolvable states of the puzzle. we assumed the filling in strategy in the way the first three boxes will be placed at the first row of the puzzle, then the boxes 4 to 6 will be placed in the second row in the order shown in figure 4.1.

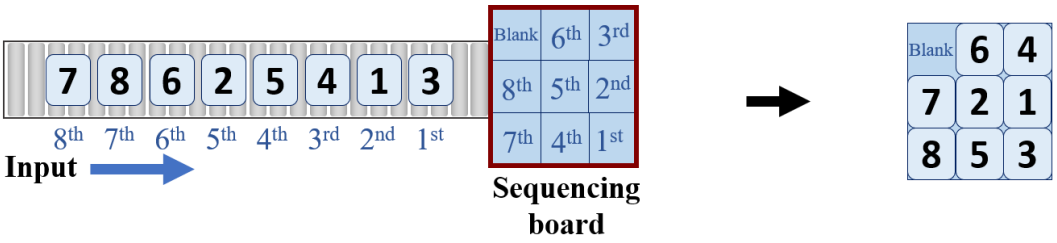


Figure 4.1 Current pre-sorting strategy

However different filling-in strategies might be applied, for instance, choosing the best configuration toward the shortest solution steps.

in this strategy, we chose the best permutation for the first three boxes which are placed in the first row, and then based on this permutation, we investigate the best permutation for the next three boxes.

Eventually, we reach the best state configuration that provides the shortest solution steps.

Figure 4.2. illustrate the concept of the new pre-sorting strategy.

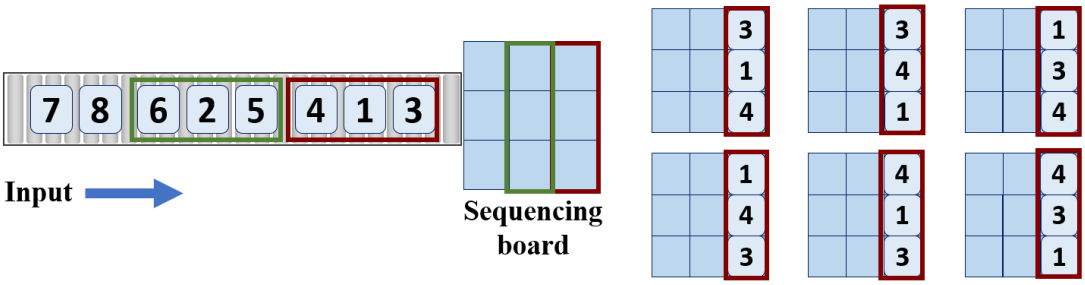


Figure 4.2. The concept of the new pre-sorting strategy.

REFERENCES

[1] Rushton Alan, Croucher Phil, B. P. The Handbook of Logistics & Distribution Management. **2010**, 636.

[2] Fumi, A.; Scarabotti, L.; Schiraldi, M. M. Minimizing Warehouse Space with a Dedicated Storage Policy. *Int. J. Eng. Bus. Manag.*, **2013**, *5* (1), 1–8. https://doi.org/10.5772/56756.

[3] Berg, J. P. V. Den; Zijm, W. H. M. Models for Warehouse Management: Classification and Examples. *Int. J. Prod. Econ.*, **1999**, *59* (1), 519–528. https://doi.org/10.1016/S0925-5273(98)00114-5.

[4] Daraei, M. Warehouse Redesign Process : A Case Study at Enics Sweden AB Master Thesis Work. *Master thesis Work*, **2014**, 1–76.

[5] Hsieh, L. F.; Tsai, L. The Optimum Design of a Warehouse System on Order Picking Efficiency. *Int. J. Adv. Manuf. Technol.*, **2006**, *28* (5–6), 626–637. https://doi.org/10.1007/s00170-004-2404-0.

[6] Gue, K. R. Very High Density Storage Systems. *IIE Trans. (Institute Ind. Eng.*, **2006**, *38* (1), 79–90. https://doi.org/10.1080/07408170500247352.

[7] Gue, K. R.; Kim, B. S. Puzzle-Based Storage Systems. **2007**, No. March 2006. https://doi.org/10.1002/nav.

[8] Shah, B.; Khanzode, V. A Comprehensive Review and Proposed Framework to Design Lean Storage and Handling Systems. *Int. J. Adv. Oper. Manag.*, **2015**, *7* (4), 274–299. https://doi.org/10.1504/IJAOM.2015.075025.

[9] Aleisa, E. E.; Lin, L. For Effective Facilities Planning: Layout Optimization Then Simulation, or Vice Versa? *Proc. - Winter Simul. Conf.*, **2005**, *2005*, 1381–1385. https://doi.org/10.1109/WSC.2005.1574401.

[10] Inman, R. R. ASRS Sizing for Recreating Automotive Assembly Sequences. *Int. J. Prod. Res.*, **2003**, *41* (5), 847–863. https://doi.org/10.1080/0020754031000069599.

[11] Gue, K. R.; Uluda, O.; Furmans, K. A High-Density System for Carton Sequencing. *6th Int. Sci. Symp. Logist.*, **2012**, No. Hodgson.

[12] Boysen, N.; Stephan, K.; Weidinger, F. Manual Order Consolidation with Put Walls: The Batched Order Bin Sequencing Problem. *EURO J.*

Transp. Logist., **2019**, *8* (2), 169–193. https://doi.org/10.1007/s13676-018-0116-0.

[13] Boysen, N.; Fedtke, S.; Weidinger, F. Optimizing Automated Sorting in Warehouses: The Minimum Order Spread Sequencing Problem. *Eur. J. Oper. Res.*, **2018**, *270* (1), 386–400. https://doi.org/10.1016/j.ejor.2018.03.026.

[14] Rethmann, J.; Wanke, E. Storage Controlled Pile-up Systems, Theoretical Foundations. *Eur. J. Oper. Res.*, **1997**, *103* (3), 515–530. https://doi.org/10.1016/S0377-2217(96)00303-7.

[15] Yalcin, A.; Koberstein, A.; Schocke, K. O. Motion and Layout Planning in a Grid-Based Early Baggage Storage System: Heuristic Algorithms and a Simulation Study. *OR Spectr.*, **2019**, *41* (3), 683–725. https://doi.org/10.1007/s00291-018-0545-z.

[16] Uriarte, C.; Asphandiar, A.; Thamer, H.; Benggolo, A.; Freitag, M. Control Strategies for Small-Scaled Conveyor Modules Enabling Highly Flexible Material Flow Systems. *Procedia CIRP*, **2019**, *79*, 433–438. https://doi.org/10.1016/j.procir.2019.02.117.

[17] Krühn, T.; Falkenberg, S.; Overmeyer, L. Decentralized Control for Small-Scaled Conveyor Modules with Cellular Automata. *2010 IEEE Int. Conf. Autom. Logist. ICAL 2010*, **2010**, 237–242. https://doi.org/10.1109/ICAL.2010.5585288.

[18] Claudio Uriarte, Hendrik Thamer, Michael Freitag, K.-D. T. Flexible Automatisierung Logistischer Prozesse Durch Modulare Roboter- Und Materialflusssysteme. **2016**, 9–14. https://doi.org/10.2195/lj_Proc_uriarte_de_201605_01.

[19] Mayer, S. H. *Development of a Completely Decentralized Control System for Modular Continuous Conveyors*; 2009.

[20] Kota, V. R.; Taylor, D.; Gue, K. R. Retrieval Time Performance in Puzzle-Based Storage Systems. *J. Manuf. Technol. Manag.*, **2015**, *26* (4), 582–602. https://doi.org/10.1108/JMTM-08-2013-0109.

[21] Gue, K. R.; Furmans, K.; Seibold, Z.; Uludag, O. GridStore: A Puzzle-Based Storage System with Decentralized Control. *IEEE Trans. Autom. Sci. Eng.*, **2014**, *11* (2), 429–438. https://doi.org/10.1109/TASE.2013.2278252.

[22] Uludaˇ, O. GridPick: A High Density Puzzle Based Order Picking System with Decentralized Control, 2014.

[23] Hao, G. ThinkIR : The University of Louisville ' s Institutional Repository GridHub : A Grid-Based , High-Density Material Handling System . **2020**.

[24] Gue, K. A High-Density , Puzzle-Based System for Rail-Rail Container Transfers. **2016**.

[25] Shekari Ashgzari, M.; Gue, K. R. A Puzzle-Based Material Handling System for Order Picking. *Int. Trans. Oper. Res.*, **2021**, *28* (4), 1821–1846. https://doi.org/10.1111/itor.12886.

[26] Yalcin, A.; Koberstein, A.; Schocke, K. O. An Optimal and a Heuristic Algorithm for the Single-Item Retrieval Problem in Puzzle-Based Storage Systems with Multiple Escorts. *Int. J. Prod. Res.*, **2019**, *57* (1), 143–165. https://doi.org/10.1080/00207543.2018.1461952.

[27] Shirazi, E.; Zolghadr, M. An Item Retrieval Algorithm in Flexible High-Density Puzzle Storage Systems. *Appl. Syst. Innov.*, **2021**, *4* (2). https://doi.org/10.3390/asi4020038.

[28] Tetouani, S.; Chouar, A.; Lmariouh, J.; Soulhi, A.; Elalami, J. A "Push-Pull" Rearrangement While Routing for a Driverless Delivery Vehicle. *Cogent Eng.*, **2019**, *6* (1), 1–14. https://doi.org/10.1080/23311916.2019.1567662.

[29] Iordan, A.-E. A Comparative Study of Three Heuristic Functions Used to Solve the 8-Puzzle. *Br. J. Math. Comput. Sci.*, **2016**, *16* (1), 1–18. https://doi.org/10.9734/bjmcs/2016/24467.

[30] Shaban, R.; Natheer Alkallak, I.; Mohamad Sulaiman, M. Genetic Algorithm to Solve Sliding Tile 8-Puzzle Problem. *J. Educ. Sci.*, **2010**, *23* (3), 145–157. https://doi.org/10.33899/edusj.2010.58405.

[31] Piltaver, R.; Luštrek, M.; Gams, M. The Pathology of Heuristic Search in the 8-Puzzle. *J. Exp. Theor. Artif. Intell.*, **2012**, *24* (1), 65–94. https://doi.org/10.1080/0952813X.2010.545997.

[32] Reinefeld, A. Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA*. *Int. Jt. Conf. Artif. Intell.*, **1993**, 248–253.

[33] Mishra, A. K.; Siddalingaswamy, P. C. Analysis of Tree Based Search Techniques for Solving 8-Puzzle Problem. *2017 Innov. Power Adv.*

Comput. Technol. i-PACT 2017, **2017**, *2017-Janua*, 1–5. https://doi.org/10.1109/IPACT.2017.8245012.

[34] J., M.; L., R.; P., S. Comparative Analysis of Search Algorithms. *Int. J. Comput. Appl.*, **2018**, *179* (50), 40–43. https://doi.org/10.5120/ijca2018917358.

[35] Nilsson, N. J. *Artificial Intelligence: A Modern Approach*; 1996; Vol. 82. https://doi.org/10.1016/0004-3702(96)00007-0.

[36] Nosrati, M.; Karimi Hojat Allah Hasanvand, R.; original, including D. Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches. *World Appl. Program.*, **2012**, *2* (24), 251–256.

[37] Zhou, Y.; Cheng, X.; Lou, X.; Fang, Z.; Ren, J. Intelligent Travel Planning System Based on A-Star Algorithm. *Proc. 2020 IEEE 4th Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2020*, **2020**, No. Itnec, 426–430. https://doi.org/10.1109/ITNEC48623.2020.9085072.

[38] Ando, R.; Takefuji, Y. A New Perspective of Paramodulation Complexity by Solving Massive 8 Puzzles. **2020**.

[39] 5x5 sliding puzzle can be solved in 205 moves https://oeis.org/search?q=5x5+sliding+puzzle+can+be+solved+in+205+moves&sort=&language=&go=Search (accessed Nov 11, 2021).

[40] Osaghae, E. O. An Alternative Solution to N-Puzzle Problem. *J. Appl. Sci. Environ. Manag.*, **2018**, *22* (8), 1199. https://doi.org/10.4314/jasem.v22i8.9.

[41] Hayes, R. The Sam Loyd 15-Puzzle. *Citeseer*, **2001**, No. June, 1–28.

[42] Parberry, I. A Real-Time Algorithm for the (N2 - 1)-Puzzle. *Inf. Process. Lett.*, **1995**, *56* (1), 23–28. https://doi.org/10.1016/0020-0190(95)00134-X.

[43] Korf, R. E.; Reid, M.; Edelkamp, S. Time Complexity of Iterative-Deepening-A*. *Artif. Intell.*, **2001**, *129* (1–2), 199–218. https://doi.org/10.1016/S0004-3702(01)00094-7.

[44] van der Werf, J. Simple Selection Theory and the Improvement of Selection Accuracy. *Anim. Breed. Use New Technol.*, **2000**, 19–34.

[45] Yaroslavskiy, V. Dual-Pivot Quicksort. **2009**, *2*, 1–11.