Master's thesis

NTNU
Norwegian University of Science and Technology

Giulia Fede

# A Puzzle-Based Movable Rack system with class-based storage policy

Master's thesis in Miscellaneous Courses - Faculty of Engineering
Supervisor: Prof. Fabio Sgarbossa
Co-supervisor: Prof. Marco Melacini (Politecnico di Milano)
July 2023

NTNU
Norwegian University of Science and Technology

POLITECNICO
MILANO 1863

Giulia Fede

# A Puzzle-Based Movable Rack system with class-based storage policy

Master's thesis in Miscellaneous Courses - Faculty of Engineering
Supervisor: Prof. Fabio Sgarbossa
Co-supervisor: Prof. Marco Melacini (Politecnico di Milano)
July 2023

Norwegian University of Science and Technology


# Abstract in English

The recent growth in e-commerce during the COVID-19 pandemic has intensified the demand for efficient warehouses with high storage density and throughput. Puzzle-based storage (PBS) solutions have demonstrated their effectiveness in optimizing storage capacity with limited spaces. To further enhance these capabilities, the Logistics 4.0 Lab at NTNU and the company Wheel.me have introduced a new system that utilizes autonomous wheels, enabling storage racks to move in any direction. This innovative approach holds the potential to significantly improve throughput capacity.

Existing studies have primarily focused on evaluating retrieval performance based on factors like escort locations, input/output (I/O) points, and movement constraints. This paper aims to enhance retrieval time by implementing a two-class-based storage policy. By strategically placing high-turnover items near the I/O point, travel distances are minimized, resulting in faster retrieval.

To evaluate the effectiveness of the class-based storage system, various input parameters are considered, including the ABC curve, system size, class A shape, and shape ratio. The objective is to examine how the optimized system with a class-based storage policy, impacts the performance of the new configuration. The findings reveal significant reductions in cycle time, with improvements of up to 150% compared to random storage, depending on the specific system configuration and characteristics of the stored items.

# Abstract in Norwegian

Den nylige veksten i netthandel under COVID-19-pandemien har intensivert etterspørselen etter effektive lagerbygninger med høy lagringskapasitet og gjennomstrømning. Puzzle-baserte lagringsløsninger (PBS) har vist seg å være effektive for å optimalisere lagringskapasiteten med begrenset plass. For å ytterligere forbedre disse evnene, har Logistikk 4.0 Lab ved NTNU og selskapet Wheel.me introdusert et nytt system som bruker autonome hjul, slik at lagringshyller kan bevege seg i alle retninger. Denne innovative tilnærmingen har potensial til å betydelig forbedre gjennomstrømningskapasiteten.

Tidligere studier har hovedsakelig fokusert på å evaluere henteytelsen basert på faktorer som eskorteposisjoner, inngangs-/utgangspunkter og bevegelsesbegrensninger. Denne artikkelen har som mål å forbedre hentetiden ved å implementere en lagringspolitikk basert på to klasser. Ved strategisk å plassere høyomsetningsvarer nær inngangs-/utgangspunktet, minimeres reiseavstander, noe som resulterer i raskere henting.

For å evaluere effektiviteten til lagringssystemet basert på klasser, vurderes ulike inngangsparametere, inkludert ABC-kurven, systemstørrelse, formen til klasse A og formforholdet. Målet er å undersøke hvordan det optimaliserte systemet med en lagringspolitikk basert på klasser påvirker ytelsen til den nye konfigurasjonen. Funnene avslører betydelige reduksjoner i syklustid, med forbedringer på opptil 150 % sammenlignet med tilfeldig lagring, avhengig av den spesifikke systemkonfigurasjonen og egenskapene til de lagrede elementene.

# Acknowledgements

I would like to thank my supervisor at Politecnico of Milano, Marco Melacini, for being responsive and providing helpful feedback. His guidance was instrumental in writing my master's thesis.
I also want to express my gratitude to my supervisor at NTNU, Fabio Sgarbossa, for guiding me in the practical work and for his willingness to schedule meetings whenever I encountered uncertainties.

I am grateful to NTNU and Politecnico of Milano for giving me the opportunity to extend my time abroad to write my master's thesis. This allowed me to fully immerse myself in the working environment at NTNU and provided me with more time to build friendships, experience Norwegian traditions, and even face the challenges of winter in Norway.

A special thank you goes to my flatmates, who were understanding, welcoming, and shared this remarkable experience with me. Each of them taught me something valuable. I am thankful to my friends from Italy for their support and closeness, and to the new friends I made during this time. We shared countless memorable moments. Being in a completely different country became an enriching and enjoyable experience, thanks to all of you.

Lastly, I want to express my deepest appreciation to my family for their support and belief in me, even when I doubted myself. You have always encouraged me to remain resilient in challenging situations and supported me in every decision I made.

# Contents

# Introduction

In recent years, the rapid growth of e-commerce has led to a greater emphasis on optimizing warehousing operations and performance. Two prominent areas of research in this field are Robotic Mobile Fulfillment (RMF) systems and Puzzle-Based Storage (PBS) systems.
However, these systems have typically been studied independently. Research on PBS systems has focused on reducing retrieval times while maintaining high storage density. On the other hand, research on RMF systems has prioritized achieving high picking rates and overall throughput capacity.

In a collaborative effort between the Logistics 4.0 Lab at NTNU (Norwegian University of Science and Technology) and the Norwegian company Wheel.me, there has been a recent conceptualization of an evolved PBS system. The goal is to combine the advantages of both PBS and RMF systems by addressing the trade-off between throughput capacity and storage density.
Previous studies have primarily concentrated on the design of the system and the development of effective movement policies. These studies have demonstrated significant improvements in performance. The specific contribution of this thesis is the adoption of a class-based storage policy, which aims to enhance the performance of the new configuration even further.

## 0.1. Motivation

The COVID-19 pandemic led to a relevant growth in online retailing, which had already been growing in popularity (Guthrie et al. 2021). This resulted in the development of new warehousing technologies suitable for e-commerce (Boysen, De Koster et al. 2019). The pandemic also exposed the vulnerability of lean sourcing and just-in-time logistics, as global supply chains were disrupted (Garnett et al. 2020). The lack of intermediate storage in supply chains further worsened the crisis, especially for critical goods like medical equipment (Bhaskar et al. 2020). The problems arising during COVID-19, in addition to

the growth of online retailing, highlighted the increasing need for compact and responsive storage systems.
Storing goods requires space, and real estate prices in urban areas have risen due to urbanization. At the same time, companies want to reduce lead times. Higher costs associated with the required storage area and rapid retrieval requirements pose a challenge.
Finally, order-picking activities in warehouses are labor-intensive and account for a significant portion of operating costs. To improve efficiency, parts-to-picker systems have been developed. These systems involve carrying inventory to a picker at a workstation, eliminating non-value-adding time spent by the pickers to move between shelves.

In collaboration with the Norwegian company Wheel.me, the Logistics 4.0 Lab at the Norwegian University of Science and Technology (NTNU) has conceptualized a new configuration based on Puzzle-Based Storage (PBS) systems that leverages the use of autonomous wheels.
Previously, no large-scale PBS system with moveable storage racks had been investigated. According to the survey conducted by Boysen, De Koster et al. (2019), PBS systems were considered unsuitable for handling the tight delivery schedules required in an e-commerce environment due to movement constraints that limited retrieval times. However, the pursuit of designing storage systems with higher densities to minimize warehousing space and reduce costs remains a prevalent topic.
This innovative approach aims to address the above-mentioned challenges in supply chain and warehouse management by maximizing storage density while maintaining high throughput levels. The new system allows for a high level of storage density and improved throughput performance by relaxing movement constraints.
The key enabler of this new configuration is the development of these autonomous wheels equipped with computational and mechatronic capabilities, allowing them to perceive their surroundings, avoid collisions, and move autonomously in any direction. Control of the wheels is achieved through a cloud computing system, allowing for the coordination of multiple objects equipped with autonomous wheels. This makes autonomous wheels a potential alternative to existing material handling equipment such as AMRs, AGVs, shuttles, and conveyor belts.
In the context of developing this new configuration of PBS systems, it is assumed that storage racks can be moved using autonomous wheels. An illustration can be found in Figure 1.

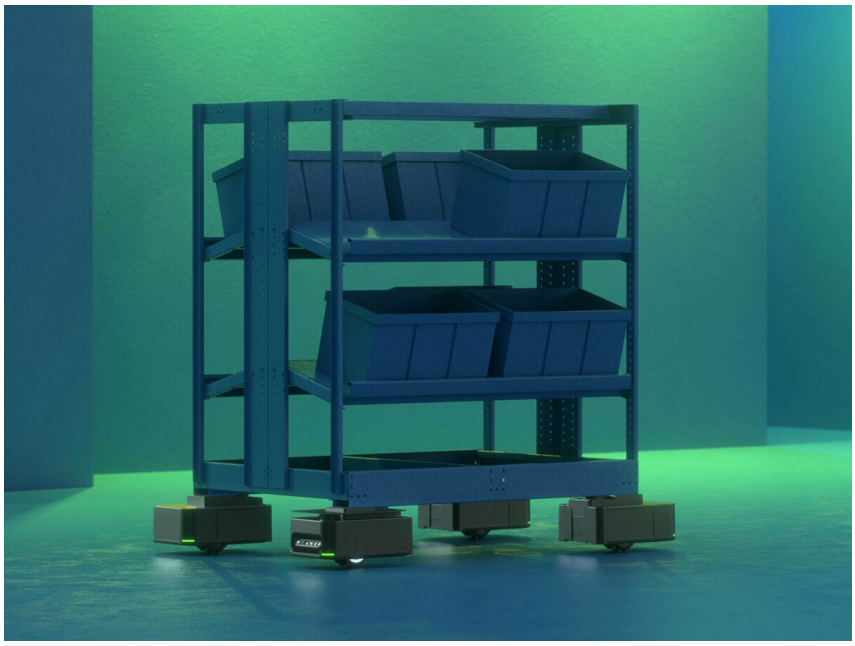


Figure 1: Autonomous wheels developed by Wheel.me

The analysis of this new system was initially undertaken by a master's student from NTNU in previous years. The objective of the study was to describe the movement policy of the system and compare its performance with other existing solutions. The findings of the student's work indicated significant improvements, highlighting the potential of the new system. Therefore, it is valuable to pursue further research on this topic and explore ways to enhance its performance even more.
Research has demonstrated the advantages associated with the implementation of a class-based storage policy in minimizing the average travel distance within a storage system. Building upon this knowledge, the primary objective of this study is to implement a two-class storage policy, optimize the system's design, and assess its impact on throughput and cycle time.

## 0.2. Research questions and method

This work has the objective to answer the following research questions:

- **RQ1:** How can we implement a class-based storage policy on the new PBS system?
- **RQ2:** How does this storage policy affect the performance of the system?
- **RQ3:** What are the optimal values for the design parameters?
- **RQ4:** What are the possible improvements and limitations of this configuration

with respect to the same system with random storage and traditional PBS systems?

Before the initialization of the work, a crucial prerequisite was to develop a comprehensive understanding of the design and load movements within the new system described in the master thesis of the previous student.
To address RQ1, an extensive review of the existing literature on class-based storage policies was conducted. The following phase consisted of the representation of the system by using Microsoft Excel (Version 2305).
The visualization of the system played a fundamental role in determining the potential configurations for class A and in adapting the analytical model to accommodate the implementation of the class-based storage policy.
To effectively answer RQ2, it became necessary to simulate the system using the Python programming language (Version 3.11). The code developed for this purpose has been validated ensuring its accuracy by comparing the results obtained with the previously constructed representation of the system in Excel. Furthermore, the Python simulation results obtained considering a random storage policy were also compared against the outcomes derived from the previous master thesis.
This simulation tool facilitated the evaluation of the system's performance across different scenarios, by varying the different input parameters. This enabled answering RQ3 through the formulation of a design procedure aimed at minimizing the average cycle time.
Lastly, RQ4 required an exhaustive review of the existing literature concerning traditional PBS systems.

## 0.3. Outline

This work is structured as follows:

- **Chapter 1: Theoretical background**
  This chapter provides an overview of the theoretical background. It introduces the PBS and RMF systems, explaining their main concepts and performance. The chapter also describes the class-based storage policy and its specific implementation in the context of this work.
- **Chapter 2: Proposed configuration**
  The focus of this chapter is on the new configuration of the system and the assumptions made for the analysis. It provides a detailed description of the movement policy implemented in the system.
- **Chapter 3: Analytical model with random storage policy**

This chapter presents the analytical model used to describe the new configuration when a random storage policy is employed.

- **Chapter 4: Analytical model with class-based storage policy**
  The analytical model developed to design and measure the performance of the system under the class-based storage policy is described in this chapter. It also discusses the choice of the shapes for the different classes.

- **Chapter 5: Design procedure**
  This chapter outlines a procedure for determining the optimal parameters for designing the system with a class-based storage policy. It provides guidelines on how to define the parameters that will yield the best performance.

- **Chapter 6: Random vs. Class-based**
  Here, the effect of implementing the class-based storage policy is examined in comparison to the random storage policy. The average cycle time is used as a measure for the assessment.

- **Chapter 7: Comparison with literature**
  The new solution is compared with existing approaches found in the literature. The purpose of this comparison is to assess the potential improvements in terms of density and average cycle time that can be achieved with the new solution.

- **Chapter 8: Conclusions and future developments**
  This chapter offers an overview of the results achieved throughout the analysis. It also discusses the limitations of the study and potential areas for future developments.

# 1 | Theoretical background

## 1.1. Puzzle-based storage

Puzzle-based storage is one of the most space-efficient storage systems. They draw inspiration from the popular 15-puzzle game, shown in Figure 1.1, where the goal is to arrange numbered tiles in a $4 \times 4$ grid by sliding them into a single empty slot.

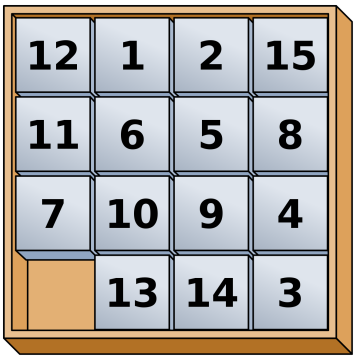



Figure 1.1: 15-puzzle game

The original puzzle-based storage (PBS) system was introduced by K. Gue and B. Kim in 2007, resembling the tile configuration but with storage units such as containers or pallets instead.
This system is represented as a grid whereby each cell can be either occupied by a load or left free of stored units. One particular cell acts as the I/O point through which loads enter or exit the system. For a desired load to be moved, the empty cell must anticipate its path. Each movement within this system is defined in relation to the empty cell and is known as an "escort move".
Figure 1.2 illustrates the series of escort moves required to transport the highlighted red load, which is the requested one, to the I/O location. Each arrow represents an escort move and they show how the escort must be moved in order to anticipate the requested load on its way to the I/O point.

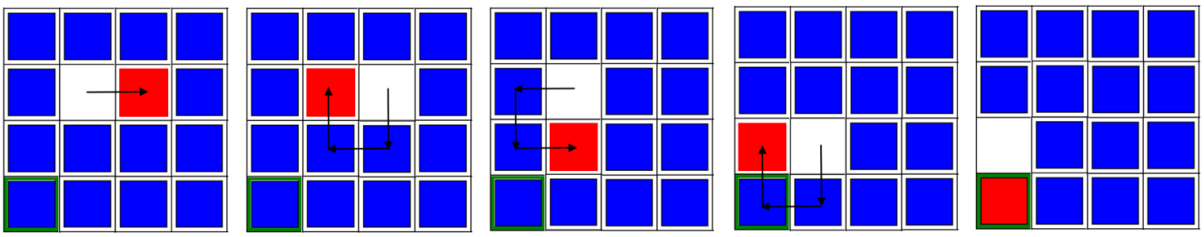


Figure 1.2: Movements in PBS

They demonstrated that a grid with n cells can theoretically achieve a density of $(n-1)/n$, extremely higher compared to the other storage systems. Only one cell, referred to as the "escort", remains unoccupied to allow the movement of storage units. They also conducted experiments to study the expected retrieval time in a PBS system and compared their findings with multi-deep aisle-based storage systems in Gue (2006). They concluded that PBS systems have the potential for high density with a reasonable trade-off in retrieval time.

This paper generated significant interest in research on PBS systems, with various objectives and configurations being investigated. The common goal was to minimize retrieval time while maintaining high density. Several researchers made contributions in this field:

- Rohit et al. (2010) used integer programming to solve optimal retrieval of a single load. However, their approach was impractical for larger grids.
- Kota et al. (2015) improved on this work by providing a closed-form expression for optimal retrieval in PBS systems with up to two escorts. They also developed a heuristic algorithm for systems with three or more escorts, which yielded near-optimal results for larger grids.
- Yalcin et al. (2019a) further improved the state of the art with an optimal algorithm called MinMov. They also developed a greedy version of MinMov as a heuristic approach for larger instances.
- Yunfeng et al. (2021) combined state-space search and beam search to reduce the number of movements and algorithmic complexity. Their work provided two heuristic contributions that reduced the computational complexity of Yalcin et al.'s (2019a) work.

All these above-mentioned works were conducted under the constraint of allowing only one escort move per step, making the minimization of load movements equivalent to minimizing retrieval time.

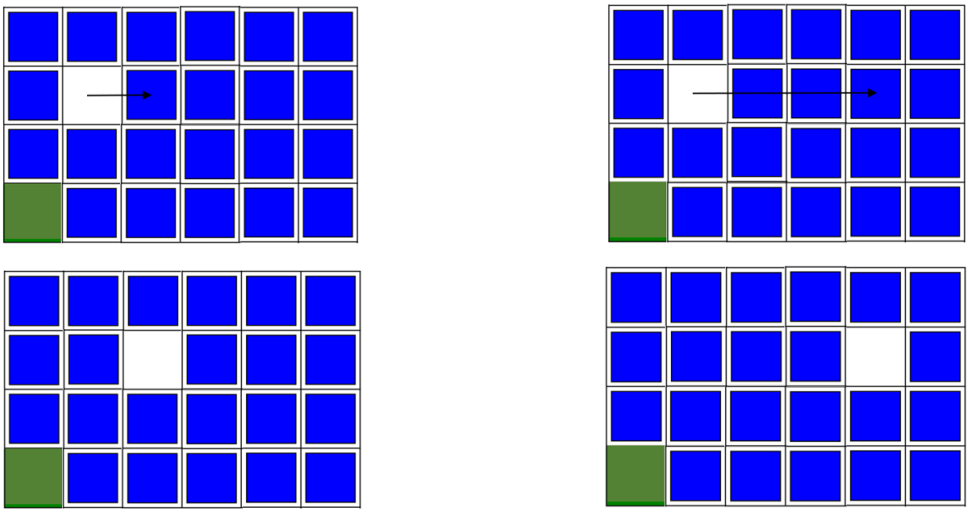
Other research was conducted allowing multiple moves to be executed within one step. In this case, the objective of minimizing the retrieval time coincides with minimizing the number of steps, instead of those of movements.

- Gue, Furmans et al. (2014) proposed a decentralized configuration of PBS systems and they showed that increasing the number of escorts per row would enhance system throughput.
- Zaerpour et al. (2017b, 2017a) proposed a multi-level PBS system called live-cube, where each level resembles earlier PBS configurations with I/O locations instead of separate input and output rows. They assumed a sufficient number of escorts to create virtual aisles for load retrieval.
- Yu et al. (2017) attempted to devise an optimal algorithm for a conventional PBS system where multiple escorts could move simultaneously. They showed that, at most, five escorts were needed to move a requested load continuously to the I/O location. However, the route taken by the load might not necessarily be the shortest distance.

Based on the extensive research conducted in the field of PBS systems, here are summarized the key concepts related to this storage system, namely the movement constraints and the creation of virtual aisles.

## Movement constraints

Research on PBS systems has highlighted the main load movement constraints. One key distinction is between single movement and block movement, illustrated in Figure 1.3.
In single movement, the escort is limited to performing just one move in a single step. This means that the escort can only move to an adjacent cell.
On the other hand, block movement allows the escort to shift to any cell within the same row or column. All the loads between the starting cell of the escort and the desired one are moved in a single step.



(a) Single movement (b) Block movement

Figure 1.3: Single vs. block movement

The choice between block movement and single movement impacts the time required for an escort to move from its starting position to a specific cell. When these cells are located in the same row or column, block movement allows the escort to make the shift in a single step. In contrast, if single movement constraints are in place, the escort would need to perform multiple steps to reach the desired cell.
Figure 1.4 demonstrates that when employing block movement, it is possible to move an escort n cells farther in a single step. In contrast, when employing single movement, the same process would require n steps.

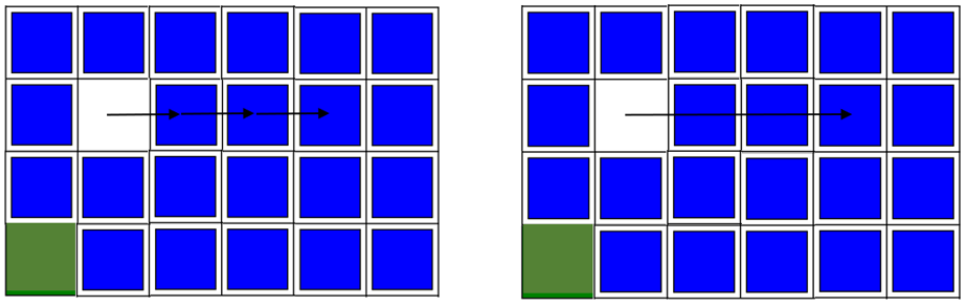


Figure 1.4: Comparison between single and block movement

This differentiation was first introduced by Gue and B. Kim (2007), who used block movements in their experiments with one escort. Yalcin et al. (2019b) investigated the relative change in average retrieval time between using load and block movement. In high-density layouts, the increase in retrieval time without block movement was less than 10%. It

was slightly higher, between 10% and 20%, with a virtual aisle strategy for similar grid configurations.

Another important distinction is between simultaneous movement, illustrated in Figure 1.5, and sequential movement.
Simultaneous movement allows for the execution of several moves in a single step. In this case, multiple escorts within the grid are allowed to move at the same time. The maximum number of simultaneous moves in a single step is equal to the number of escorts present in the system.
Sequential movement, on the other hand, limits the movement to one move per step. This means that each escort is moved individually in separate steps.

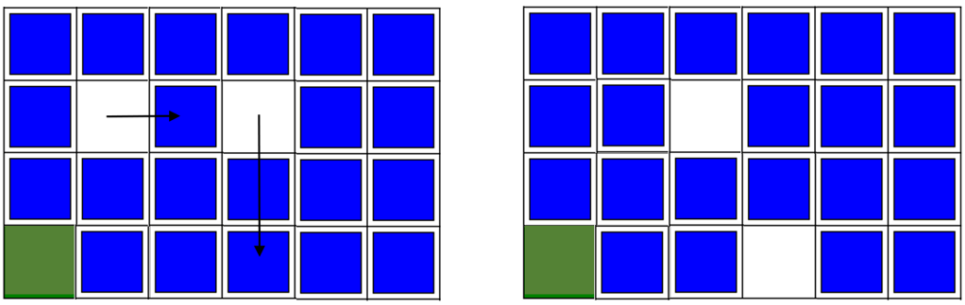


Figure 1.5: Simultaneous movement

Note that simultaneous movement should not be confused with block movement. Block movement involves moving multiple loads with a single escort, whereas simultaneous movement refers to moving multiple escorts simultaneously. Consequently, simultaneous movement allows performing multiple block moves at the same time.

Yu et al. (2017) investigate the effect of simultaneous movement compared to sequential movement in PBS systems. Their finding was a 50% improvement in retrieval time compared to the state-of-the-art retrieval algorithm at the time by Kota et al. (2015).
In general, research on PBS systems acknowledges that the complexity of these systems increases as the number of escorts involved grows (Gue and B. Kim, 2007; Mirzaei et al., 2017; Yalcin et al., 2019a; Yunfeng et al., 2021). This complexity arises particularly when simultaneous movement and block movement are allowed, as conflicting moves between escorts can occur (Bukchin and Raviv, 2020).
Studies focusing on minimizing retrieval times in PBS systems with high storage density tend to avoid this issue by not allowing simultaneous movement (Yunfeng et al., 2021).

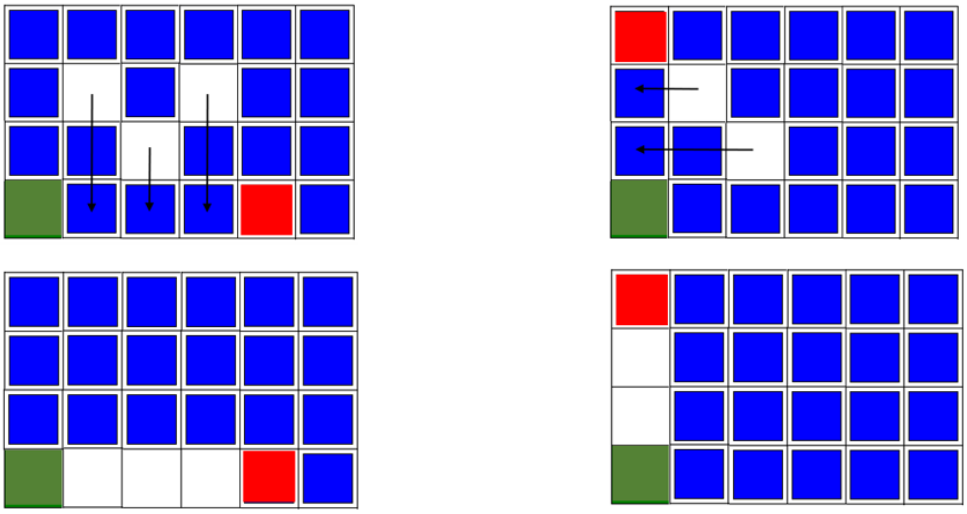
## Virtual aisles

The concept of virtual aisles in PBS systems is a strategic approach that facilitates the uninterrupted movement of requested loads toward the I/O location. It involves strategically positioning escorts in cells along the path that the requested load needs to travel. By doing so, a virtual aisle is created, eliminating the need for physical aisles typically found in traditional storage systems (Gue, Furmans et al., 2014).

In PBS systems employing the virtual aisle strategy, loads can be dynamically moved within the grid with the aim to open an aisle in front of the requested load. Virtual aisles can be created in both cardinal directions (horizontal and vertical) and potentially even diagonally, enhancing the flexibility of the system.

The implementation of virtual aisles enables higher storage density since only enough empty space for a single aisle is required. Although this slightly reduces density compared to traditional systems, it allows for faster load retrieval.

Figure 1.6 illustrates the block moves necessary to open a virtual aisle in front of a requested load, which is highlighted in red. Depending on the position of the requested load, the figure shows how to create either a horizontal or a vertical virtual aisle.



(a) Horizontal aisle (b) Vertical aisle

Figure 1.6: Opening of virtual aisles

Research by Gue, Furmans et al. (2014) and Yalcin et al. (2019b) explores different configurations and strategies in the context of baggage handling. Simulation results show that virtual aisle configurations can achieve densities up to 89% and average retrieval times of 18.5 seconds, while traditional puzzle-based strategies achieve a density of 99% with an average retrieval time of 22.7 seconds.

It is important to highlight that the choice of movement constraints impacts the efficiency of virtual aisle strategies. Not allowing for simultaneous movement results in a slight advantage of only 4.2% in average retrieval time compared to traditional puzzle-based strategies. However, the creation of virtual aisles requires approximately 28% more load movements.

## 1.2. Robotic mobile fulfillment systems

Robotic fulfillment (RMF) systems are part-to-picker systems designed to facilitate the picking process. They use autonomous mobile robots to bring storage racks to workstations where workers pick items or restock shelves.
These autonomous robots can travel underneath the racks when empty, but they require aisles when carrying racks.

The RFS was first proposed as Kiva Systems by Wurman et al. (2008). It is now known as Amazon Robotics after its acquisition in 2012 and it is currently employed in many Amazon fulfillment centers in the US and Europe.


Figure 1.7: Amazon Robotics

Using robots to carry racks has proven to be much faster than manual picking. The robots can process 200 to 300 order lines per hour, while humans can only manage 50 to 100.

RMF systems are particularly suitable for environments with a large number of small stock-keeping units (SKUs) and relatively low order volumes. They are also effective in handling fluctuations in demand, which is common in e-commerce warehouses (Boysen, De Koster et al., 2019).

Most of the research on robotic mobile fulfillment systems focuses on operation planning and control, including topics such as order and rack sequencing, storage policies, workstation assignment, and robot path planning.
Order and rack sequencing research aims to investigate an efficient assignment of orders and racks to workstations (Boysen, Briskorn et al. (2017); Merschformann et al.(2019); H. Kim et al. (2020); Xie et al. (2021)). Storage policies research focuses on the efficient division of the stock-keeping units (SKUs) among racks and the assignment of racks to storage zones (Merschformann et al. (2019); H. Kim et al. (2020); T. et al. (2020)). Workstation assignment research aims at optimizing performance by assigning workstations to robots (Zou et al. 2017). Robot path planning research tries to minimize robot travel time and maximize robot utilization (Gharehgozli and Zaerpour 2020) but also avoid conflicts between robots (Keunget al. 2020).

Some recent papers also explore system design optimization problems. Lamballais et al. (2017) investigated aspects like the length-to-width ratio of the storage area and the number of workstations. They found that the placement of workstations had a significant impact on throughput. Moreover, using a semi-open queueing network, they estimated that in a system with 14 aisles and 12 cross aisles, the maximum throughput could reach 1172 orders per hour when 14 AMRs are used with five workstations.

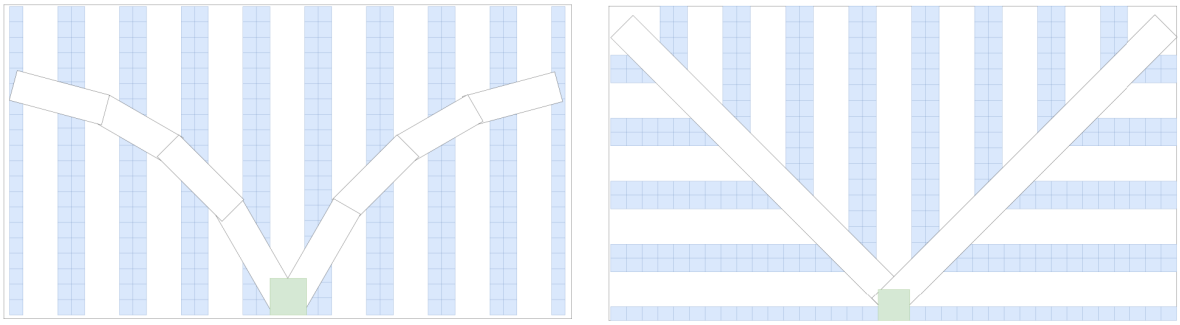
P. Yang et al. (2021) conducted research on a multi-deep RMF system, aiming to improve density while maintaining high throughput. In this configuration, storage racks might need rearrangement before being transported to a workstation. Using a semi-open queuing network, they estimated a throughput of 334 units per hour and a density of 50% with three workstations and 14 robots. They compared the result with a single-deep configuration that allows a density of 46%. They pointed out that increasing the deepness of the storage blocks to increase the density of the system came with a high cost in throughput.

Wang et al. (2020) also explored multi-deep configurations in RMF systems, focusing on the travel time perspective of robots. They concluded that an optimal depth for storage blocks is 2, as travel time increases significantly with greater depth.

The travel distance of loaded robots is constrained by the aisle configuration, being the robots required to travel through the aisles when transporting racks. To minimize the travel distance of loaded robots, X. Yang et al. (2021) introduced diagonal cross aisles as a design optimization perspective.

## 1.3. Non-traditional aisle configurations

Gue and Meller (2009) challenged traditional warehouse design rules of parallel picking aisles and orthogonal cross aisles by proposing two alternative aisle configurations to minimize the travel distance: the flying V and fishbone configurations.
These configurations are shown in Figure 1.8. In the flying V configuration, they allow for non-orthogonal and non-straight cross aisles, while in the fishbone configuration, they allow for non-parallel picking aisles.



(a) Flying V configuration (b) Fishbone configuration

Figure 1.8: Non-traditional aisle configurations

Their analytical model showed that these alternative configurations can reduce the expected travel distance with respect to the traditional aisle configuration respectively by 8% - 12% for the flying V configuration and 10% - 20% for the fishbone configuration.
Because of the diagonal and more direct path of pickers, these configurations allow for a travel distance closer to the Euclidean distance rather than the Manhattan one, imposed by the traditional configuration.
They also presented a theoretical lower bound for travel distance, obtained by avoiding the constraint imposed by the aisles and considering the travel distance equal to the Euclidean one. They found that the maximum theoretical reduction in the expected travel distance is 23.5%, which indicated that the fishbone configuration is almost optimal.

Bortolini et al. (2015) further investigated configurations with multi-diagonal cross aisles and parallel picking aisles with the aim to minimize the travel distance. They identify the possible areas of application.
A configuration with two diagonal cross aisles allows a reduction of the average travel distance by 7% - 11.5% for smaller warehouses (under $10.000m^2$. A configuration with four diagonal cross aisles results in a reduction of the travel distance by 11.5% - 15% for medium-sized warehouses (between $10.000m^2$ and $65.000m^2$). Larger warehouses (over $65.000m^2$) could achieve a reduction of 15% - 17% with six diagonal cross aisles.

X. Yang et al. (2021) investigated the impact of a flying V configuration in a single-deep RMF system with the aim to minimize robot travel distance. They found that the average robot travel distance could be reduced by approximately 9% - 18% compared to a traditional RMF layout.

## 1.4. Class-based storage policy

The class-based storage policy is widely acknowledged as the prevailing approach in practical storage systems. Its fundamental principle involves categorizing items to be stored into a limited number of classes based on their demand rates.

Extensive literature discussions (De Koster et al., 2007; Graves et al., 1977; Gu et al., 2007; Hausman et al., 1976; Rosenblatt and Eynan, 1989; Thonemann and Brandeau, 1998) consistently demonstrate its effectiveness in reducing travel distance as well as storage and retrieval time.
Hausman et al., 1976 demonstrated that a class-based storage policy with two classes allows for a percentage improvement over a random storage assignment that ranges between 18% for an ABC curve 60/20 and 53% for an ABC curve 90/20.
Bartolini et al. (2018) demonstrated the positive outcomes achieved by implementing a class-based storage policy with two and three classes in a storage system featuring diagonal cross aisles. The savings in travel time ranged from 28.6% to 32.9%.

In the forthcoming chapters, the class-based storage policy, derived from the renowned ABC curve, is applied to the proposed system, utilizing two distinct classes. Specifically, a smaller yet more frequently requested set of loads is grouped into class A, while the remaining loads are allocated to class B. Class A racks are strategically positioned closest to the I/O point, and a randomized storage policy is implemented within each class to

optimize efficiency.

## ABC curve

The ABC curve represents a mathematical function that maps the cumulative demand of items belonging to a specific class:

$$F(x) = \frac{(1+s)x}{s+x} \tag{1.1}$$

where $x$ is the cumulative fraction of the total storage space and $s$ denotes the shape factor.
For the purpose of this work, this function has been employed to estimate the proportion of loads belonging to class A in relation to the overall number of loads requested. Consequently, $x$ represents the cumulative fraction of storage racks dedicated to class A items.

Different values of the shape factor can be considered to define distinct ABC curves, as it is shown in Table 1.1.

| Shape factor (s) | ABC curve |
|---|---|
| 1.39 | 30/20 |
| 0.60 | 40/20 |
| 0.33 | 50/20 |
| 0.20 | 60/20 |
| 0.12 | 70/20 |
| 0.07 | 80/20 |
| 0.03 | 90/20 |

Table 1.1: ABC curves and shape factor

The notation 80/20 means that the curve adheres to the Pareto principle, indicating that 80% of the overall requested loads are stored in only 20% of the storage racks. Similar interpretations can be made for the other curves.

Observing Figure 1.9, it becomes evident that as the shape factor decreases, the curve exhibits a greater degree of skewness. In other words, the distribution becomes more

imbalanced, with a higher concentration of requested loads in a specific portion of the storage space.

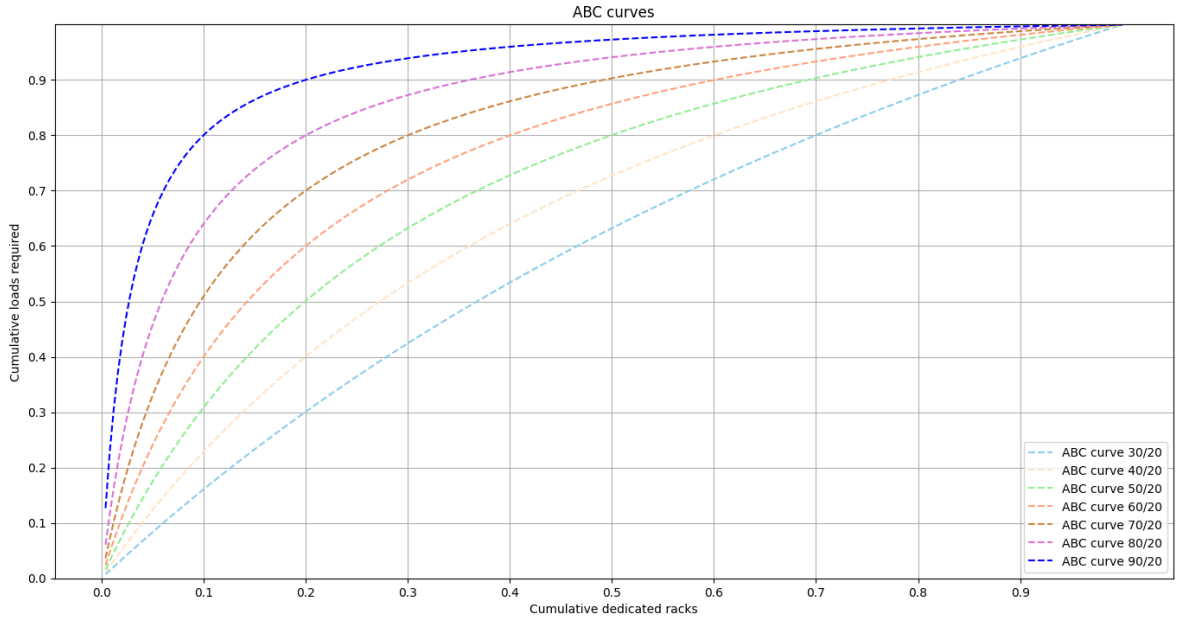



Figure 1.9: Different ABC curves

Several values for the percentage of racks dedicated to class A ($x$) have been selected and the respective probability that a generic requested load belongs to class A ($F(x)$) has been computed using Formula 1.1 for different ABC curves. The outcomes of these computations are displayed in Table 1.2.

| x | F(x) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 30/20 | 40/20 | 50/20 | 60/20 | 70/20 | 80/20 | 90/20 |
| 0.05 | 0.083 | 0.123 | 0.174 | 0.240 | 0.329 | 0.457 | 0.655 |
| 0.1 | 0.160 | 0.229 | 0.308 | 0.400 | 0.509 | 0.640 | 0.800 |
| 0.15 | 0.233 | 0.320 | 0.415 | 0.514 | 0.622 | 0.738 | 0.864 |
| 0.2 | 0.300 | 0.400 | 0.500 | 0.600 | 0.700 | 0.800 | 0.900 |
| 0.25 | 0.364 | 0.471 | 0.572 | 0.667 | 0.757 | 0.842 | 0.923 |
| 0.3 | 0.424 | 0.533 | 0.632 | 0.720 | 0.800 | 0.873 | 0.939 |
| 0.4 | 0.534 | 0.640 | 0.728 | 0.800 | 0.862 | 0.914 | 0.960 |
| 0.5 | 0.632 | 0.727 | 0.800 | 0.857 | 0.903 | 0.941 | 0.973 |

Table 1.2: $F(x)$ for different ABC curves

An intuitive example is provided. Assuming an ABC curve with the notation 60/20, if 30% of the total storage racks are dedicated to class A items, then the probability that a load belonging to class A is requested would be 80%.

Moving forward, it is important to note that in the subsequent chapters, the fraction of storage racks assigned to class A will be denoted as $l_A$, while the corresponding probability will be referred to as $p_A$.

# 2 | Proposed configuration

The proposed system is a PBS system with a unique configuration. In this system, the storage racks are equipped with autonomous wheels, enabling them to move within the facility. What sets this configuration apart is that the storage racks have the ability to move diagonally, which is not commonly considered in traditional research approaches on PBS, primarily considering horizontal and vertical movements.
Section 2.1 outlines the underlying assumptions that have been considered in the system's design. Section 2.2 provides a comprehensive description of the system itself, including the movement policy governing the relocation of storage racks. The specific movement policy adopted within a square subgrid is presented in Section 2.3, extending then the analysis to a rectangular subgrid in Section 2.4.

## 2.1. Assumptions

The overall analysis in this work is founded on several practical assumptions about the system design, movement policy, and implementation of the class-based storage policy.

Regarding the overall system design, the following key assumptions have been made:

1. The storage area is represented as a discrete grid with cell dimensions of 1m $\times$ 1m.
2. Each load resembles one storage rack which can move with autonomous wheels.
3. A load can be moved with a velocity v = 1m/s
4. The time required to change direction is neglected.
5. Single command operations, commonly termed single load retrieval in PBS systems literature, are assumed. This means that a new load can only be requested once the previous load has arrived at the I/O location.
6. When a load arrives at the I/O location, a load ready for immediate replenishment into the grid is available.

Regarding the movement policy, the following assumptions have been made:

1. An escort move involves relocating an escort from its initial cell to a new cell by moving the loads between the two locations. Each load movement is a consequence of an escort move.
2. From any given cell, a load can be moved to any of its eight adjacent cells. This includes four adjacent cells in the same row or column, as well as four adjacent cells diagonally.

Lastly, assumptions have been introduced for the design of the class-based storage policy:

1. Loads within each class are requested based on a uniform distribution being the random-based storage policy implemented within each class.
2. Items are classified using the ABC curve approach.
3. The required storage space for items is equivalent to their average inventory levels. Further details regarding this assumption will be provided in section 8.3.

## 2.2. Movement policy

In this chapter, a description of how the loads move in the system is provided.
The following convention will be followed through this work. A general rectangular grid is defined with dimensions $m \times n$ cells, where m and n always represent the number of cells along the shorter and longer sides of the grid, respectively.
The I/O location is consistently positioned in one of the corners of the grid, with coordinates (0, 0). Each cell is defined to have positive coordinates (i, j) with respect to the I/O point.

### Travel route

A route consists of the cells the requested load needs to traverse in order to reach the I/O point from its initial location. The length of the route is directly determined by the number of cells included within the route itself.
Because the loads are free to move to any adjacent cells, including those not necessarily in the same row or column, the distance between two points is determined by the Chebyshev distance. It allows for the reduction of the route length with respect to the Manhattan distance, which is constrained to rectilinear moves only. In a 2-dimensional grid, the

Chebyshev distance ($d_C$) between two points with Cartesian coordinates $(X_1, Y_1)$ and $(X_2, Y_2)$ is defined as the maximum difference between the coordinates of the two points:

$$d_C = max\ \big(\ |X_1 - X_2|,\ |Y_1 - Y_2|\ \big) \tag{2.1}$$

As a result of the convention introduced above, the Chebyshev distance between a cell and the I/O location can be simplified as:

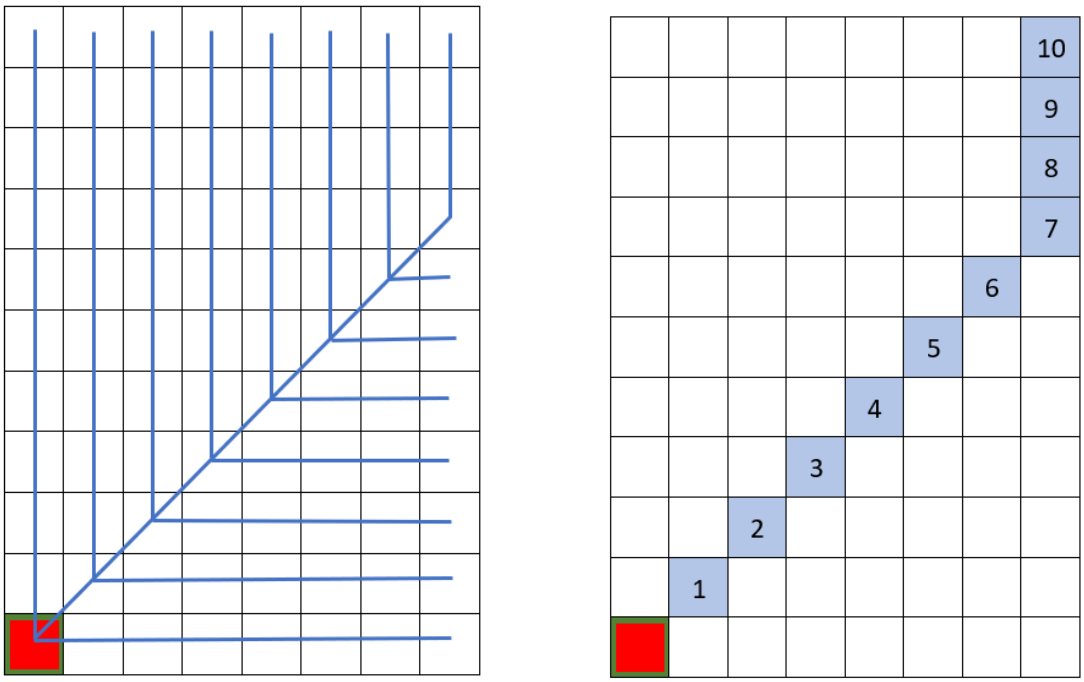
$$d_C = max\ (i; j) \tag{2.2}$$

Figure 2.1 shows an example of a grid where the value of the Chebyshev distance to reach the I/O point is computed for each load. The route length for the purpose of this work is assumed to be equal to the Chebyshev distance.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 7 |
| 5 | 5 | 5 | 5 | 5 | 5 | 6 | 7 |
| 4 | 4 | 4 | 4 | 4 | 5 | 6 | 7 |
| 3 | 3 | 3 | 3 | 4 | 5 | 6 | 7 |
| 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Figure 2.1: Chebyshev distance

There are several routes from a generic load to the I/O point whose length is equal to the Chebyshev distance. The route taken into account for our purpose is the one composed of a rectilinear path and a diagonal path. This route is shown in Figure 2.2.a. The reason for the choice is that it resembles the route that loads would travel in the fishbone configuration presented in section 1.3.




(a) Travel route for each load in the grid (b) Example travel route for a specific load

Figure 2.2: Travel route

It is easy to understand that, given a generic load of coordinates $(i; j)$, the length of the rectilinear path is equal to $max\ (i; j) - min\ (i; j)$, while the length of the diagonal path is equal to $min\ (i; j)$. A practical example is shown in Figure 2.2.b where the load has coordinates $(7; 10)$. The length of the rectilinear path is $10 - 7 = 3$ cells, while the length of the diagonal path is equal to 7 cells.

## Virtual aisle

The use of virtual aisles in this configuration allows the loads to travel from their starting position to the I/O point without any interruptions. Given the travel route followed by the loads, the virtual aisle is strategically designed to align with this route.
Similar to what has already been done with respect to the travel route, it is convenient to divide the concept of a virtual aisle into a rectilinear aisle and a diagonal aisle.

Diagonal aisles are necessary when loads need to move diagonally. Each cell along the diagonal path, except for the I/O location, is assigned an escort. The diagonal movement of loads requires additional escorts, which do not have the function of traditional escorts, but they are required to allow the loads to pass through the diagonal path. They are initially placed in the two adjacent cells in the same row or column as each cell on the

diagonal path, except for the two cells at the ends of the diagonal path, which require only one adjacent escort.
Rectilinear aisles, on the other hand, are described in the existing literature on PBS systems. To ensure uninterrupted movement, a rectilinear aisle of the same length as the rectilinear path of a requested load's route is created. The number of escorts needed to open a rectilinear aisle is equal to the length of the rectilinear path.

In a grid with dimensions $m \times n$, $m$ escorts are placed along the diagonal path to establish the diagonal aisle. Theoretically, $n$ escorts are required to create the rectilinear aisle, as the rectilinear path can be as long as the longest side of the grid. However, since the diagonal path coincides with a movement of the load in the rectilinear direction by $m$ cells, the actual number of escorts strictly required for the rectilinear path is equal to $n-m$. These escorts are placed along the longest side of the grid, connecting the diagonal aisle and rectilinear aisle. It should be noted that, based on the system assumptions, the rectilinear aisle can be opened in just one step.
Figure 2.3 illustrates the initial allocation of escorts in a generic grid with dimensions $m \times n$.

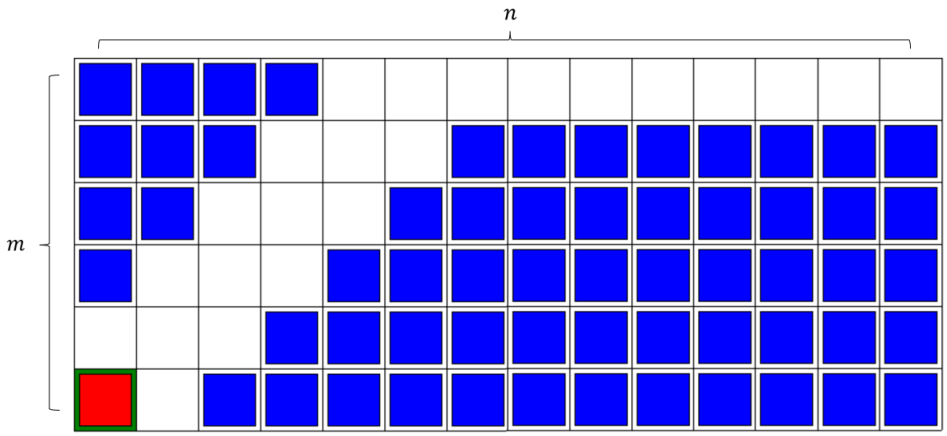



Figure 2.3: Initial allocation of escorts

To simplify the understanding of escort allocation and movement policies, the overall grid is divided into a square sub-grid and a rectangular sub-grid.
The square sub-grid has dimensions $m \times m$ and contains the I/O location, while the rectangular sub-grid has dimensions of $m \times (n - m)$. This division is shown in Figure 2.4 and it is particularly useful when dealing with grids that are not square in shape. A rectangular grid of whatever dimensions can always be modeled as a combination of these subgrids.

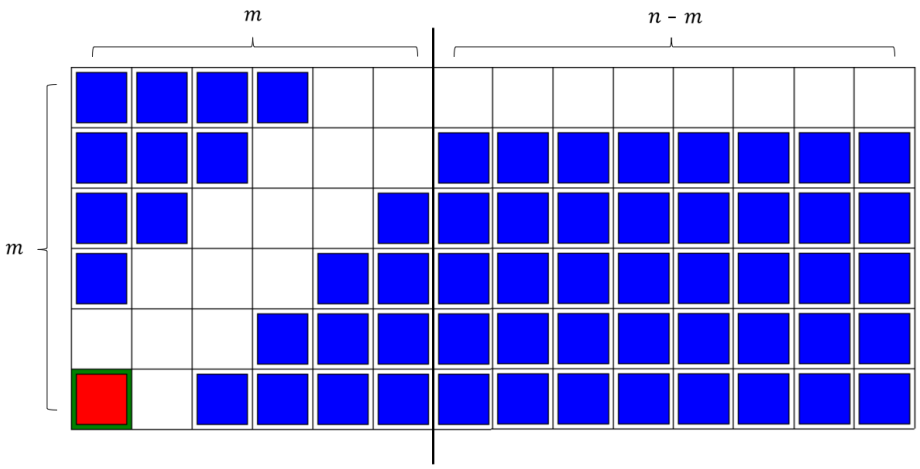



Figure 2.4: Division of the grid

Regardless of the shape and dimensions of the overall grid, the square sub-grid always exists and serves as the foundation for the analysis. The escort configuration within this sub-grid follows a unique approach, different from any other previously proposed methods, because of the presence of the diagonal aisle.
The rectangular sub-grid represents the remaining portion of the overall grid that extends beyond the square sub-grid. While it is typically in the form of a rectangle, it's important to note that even if this sub-grid has a square shape, it is still referred to and modeled as the rectangular sub-grid since it does not contain the I/O location. This ensures consistency in terminology and modeling conventions throughout the analysis.

## 2.3. Moviment policy for a square grid

A square grid can be modeled simply by considering the square sub-grid. The movement policy for retrieving a load in the square sub-grid involves three main tasks: opening and closing a rectilinear aisle, moving the requested load, and preparing the grid for the next request (referred to as cycling). To accomplish these tasks, moves are categorized into three types: rectilinear aisle moves, requested load moves, and cycle moves.

### Rectilinear aisle move

A rectilinear aisle move (RAM) is responsible for either opening or closing a rectilinear aisle. In the square sub-grid, opening an aisle involves moving escorts from the diagonal aisle to the cells in front of the requested load. Remember that each cell on the diagonal path has two escorts on either side.

Each load of the grid is located on one side of the diagonal aisle. Escorts adjacent to the diagonal path and on the same side of the load are moved with a block movement to create a rectilinear aisle.
Each RAM coincides with the block movement performed by each of these escorts. By performing all the required RAMs simultaneously, the opening of the rectilinear aisle can be done in just one step. The opening of a rectilinear aisle is shown in Figure 2.5.

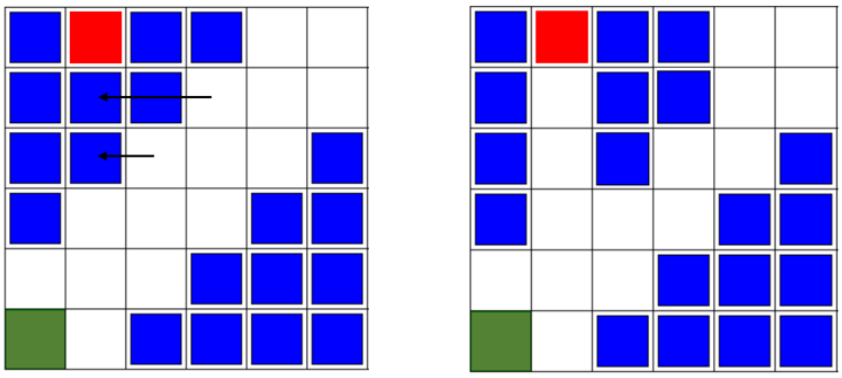
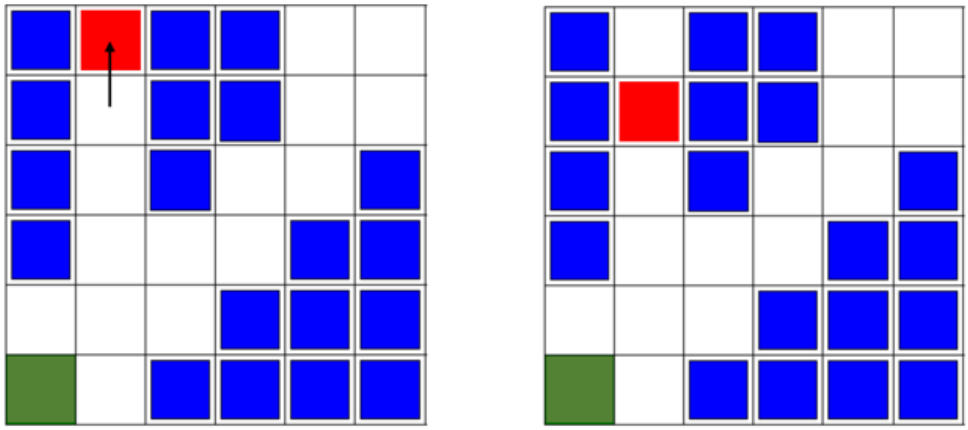


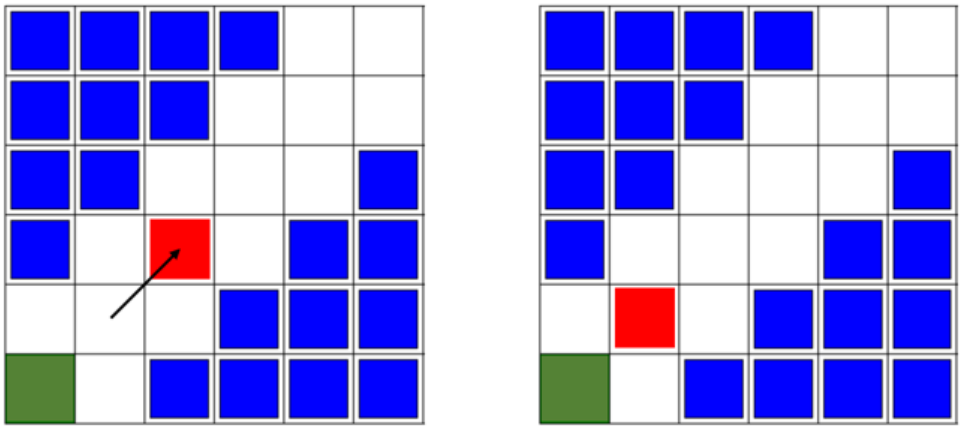
Figure 2.5: Opening of a rectilinear aisle with two RAMs

## Requested load move

A requested load move (RLM) is responsible for moving the requested load to a cell closer to the I/O point.
To execute any RLM, there must be an escort preceding the requested load. Therefore, if the rectilinear path is longer than two cells, a rectilinear aisle must be created before the load can move. This means that a RAM must be performed before the first RLM.
RLMs are always single movements, but both cycle moves and RAMs aimed at closing the rectilinear aisle can be made simultaneously to them.
Figure 2.6 illustrates how RLMs are performed in both parts of the travel route.



(a) RLM on rectilinear path



(b) RLM on diagonal path

Figure 2.6: RLMs performed in different directions

## Cycle move

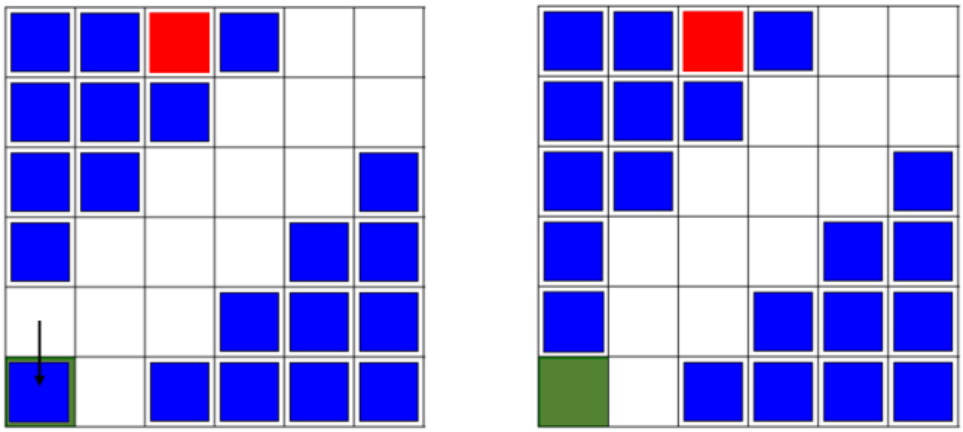
Cycle moves (CMs) are responsible for restoring the initial configuration of the grid when a new load is requested. Overall, CMs ensure that the diagonal aisle and the I/O location are clear of loads and that loads are replenished back in the grid. For each requested load, three CMs are required.

When a new load is requested, the load to be replenished is currently in the I/O location. Therefore, the first CM is performed to clear the I/O location by moving one of the adjacent escorts. There are two scenarios:

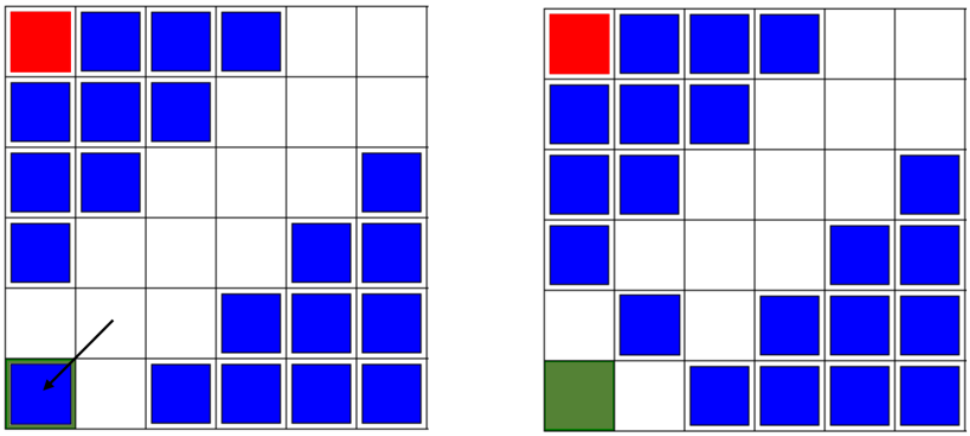
1. If the new requested load is in the same row or column as the I/O location, an escort moves from the diagonally adjacent cell to the I/O location.

2. If the new requested load is not in the same row or column, an escort moves from one of the two rectilinearly adjacent cells on the same side of the diagonal path as the I/O location.

Figure 2.7 illustrates the first CM in both scenarios.



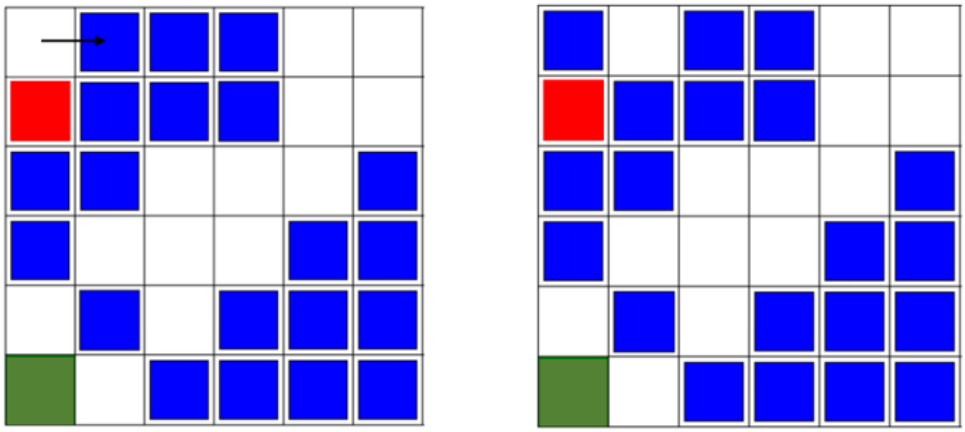
(a) First CM in the first scenario



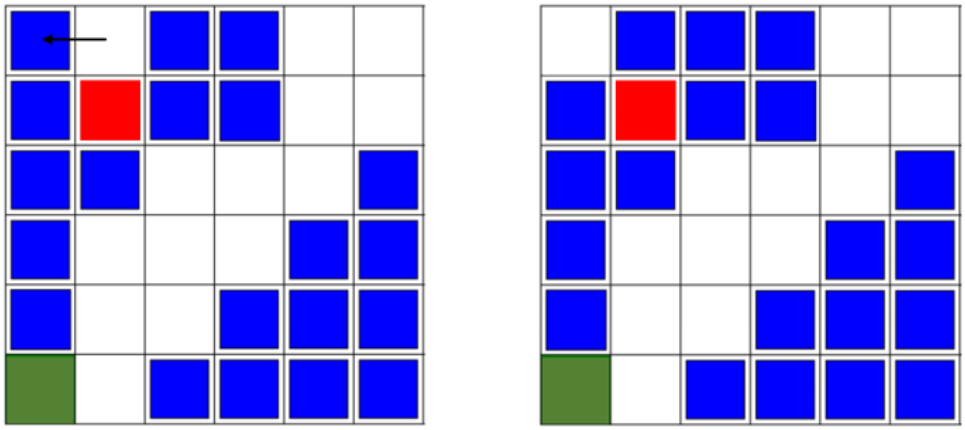
(b) First CM in the second scenario

Figure 2.7: First CM performed in both scenarios

The second CM is a block move from the initial cell of the requested load to the cell in the same row or column where the load occupying the I/O location was moved in the previous CM.
Figure 2.8 shows how the second CM is performed for both scenarios. Note that the first RLM and the first RAM to open the rectilinear aisle have been performed simultaneously with the first CM. Therefore, the initial cell of the requested rack is occupied by an escort when the second CM is performed.



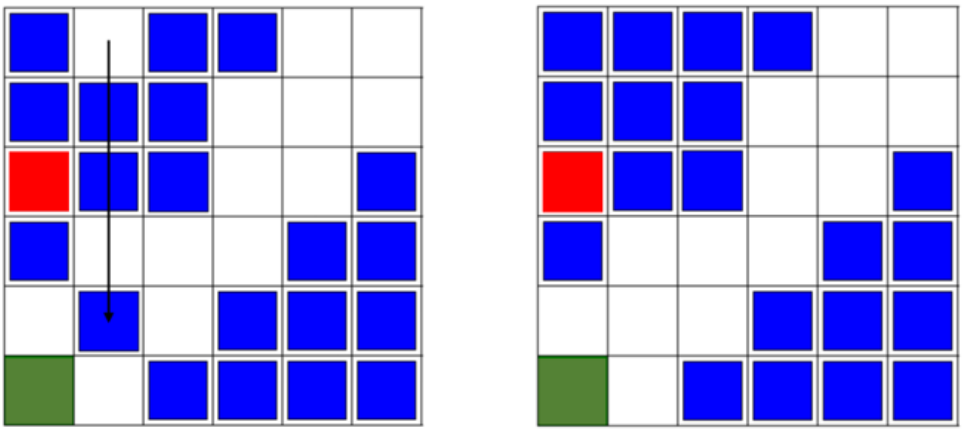
(a) Second CM in the first scenario



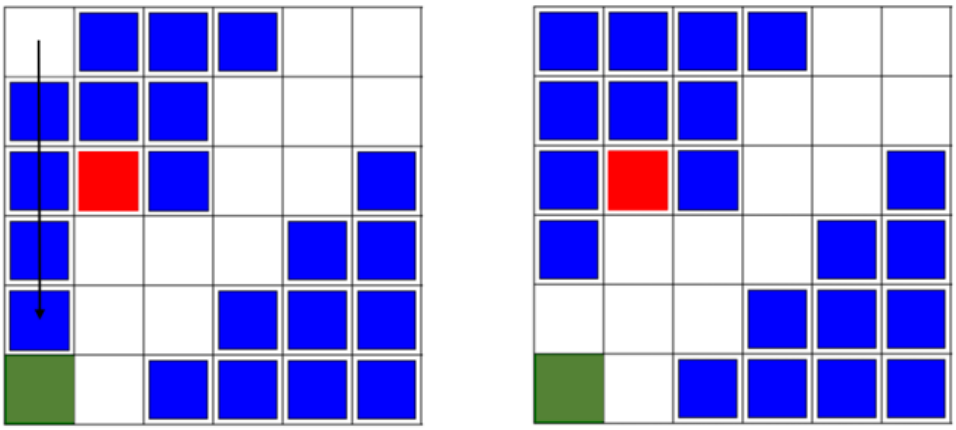
(b) Second CM in the second scenario

Figure 2.8: Second CM performed in both scenarios

The last CM performed considering the same escort moved in the second CM. It consists of a block move to move rectilinearly the escort to the cell adjacent to the I/O location. Figure 2.9 shows this third CM for requested loads in both scenarios. Note that, the second RLM, the second RAM to open the rectilinear aisle, and the first RAM for closing it have been performed during the second CM.



(a) Third CM in the first scenario



(b) Third CM in the second scenario

Figure 2.9: Third CM performed in both scenarios

These three CMs must be executed sequentially. However, as already highlighted, each CM can be performed simultaneously with both RAMs and RLMs.

Figure 2.10 shows the sequence of all the movements performed for the retrieval of a generic requested load.

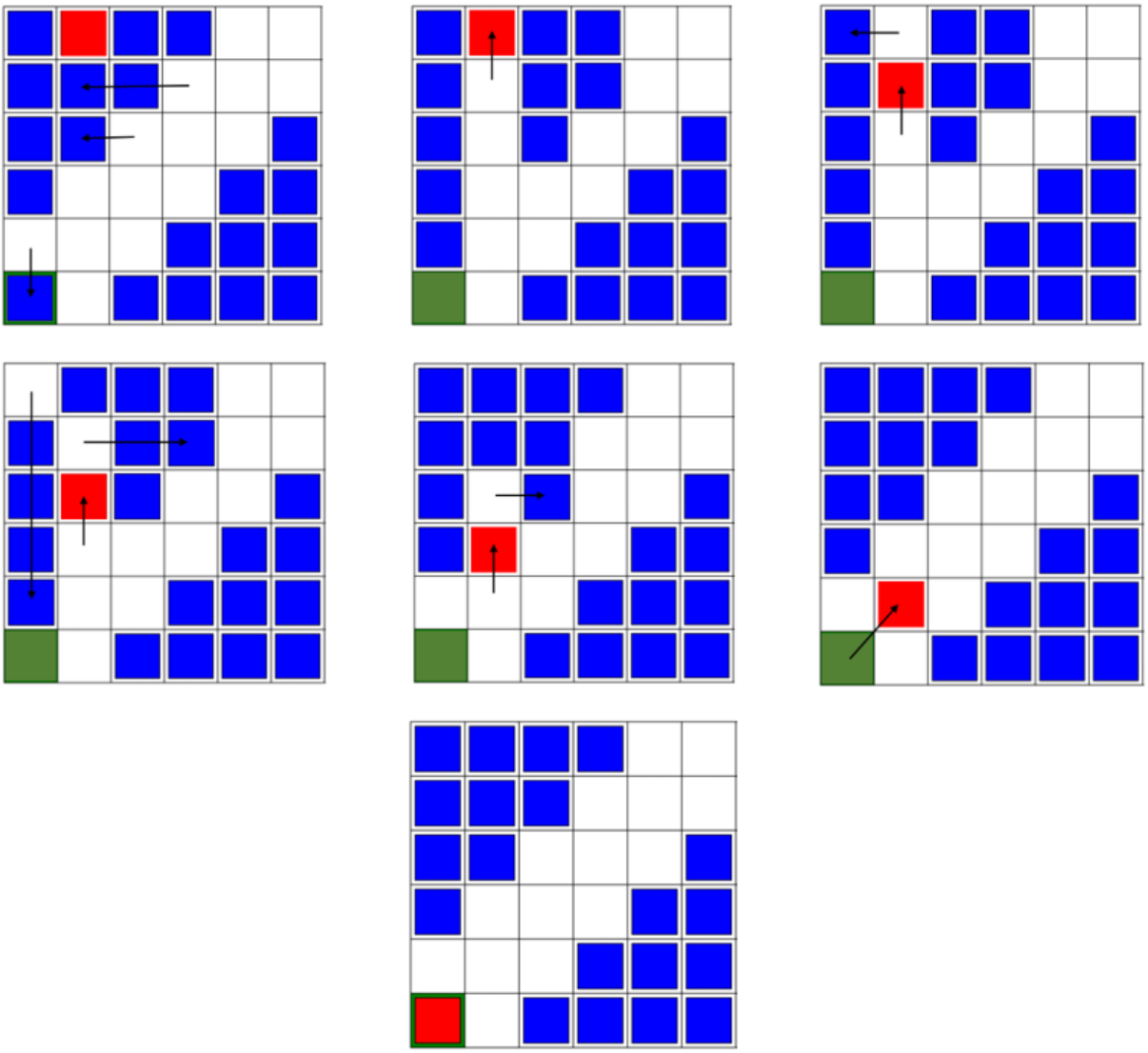


Figure 2.10: Movements required in a square sub-grid

## 2.4. Moviment policy for a rectangular grid

In the rectangular sub-grid, loads only need to travel through a rectilinear aisle. Similarly to the square sub-grid, the chosen movement policy ensures that for any cell there is always an escort that can move to this cell in one step.
Initially, escorts are located on the upper or bottom part of the sub-grid. When a load is requested, a rectilinear aisle must be opened in the cells between the requested load and the square sub-grid. All the RAMs needed to open a rectilinear aisle can be performed simultaneously in one step as illustrated in Figure 2.11.

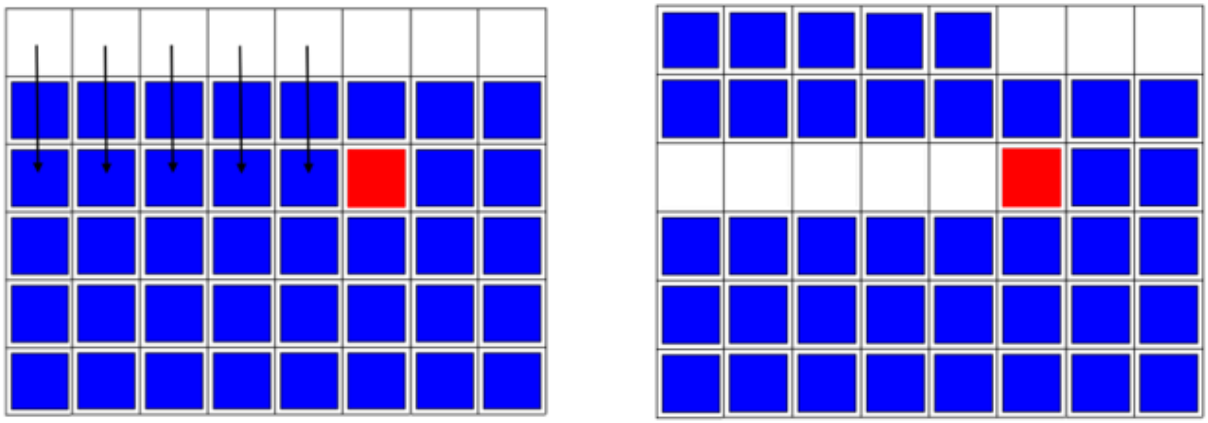


Figure 2.11: Opening of a rectilinear aisle in the rectangular sub-grid

Note that for the next requested load, the escorts will just move from the previous rectilinear aisle to the new rectilinear aisle. Therefore RAMs for closing the rectilinear aisle that characterize the square sub-grid are not necessary
The load is moved toward the square sub-grid only performing RMLs on a rectilinear path. Figure 2.12 shows a general RLM in the rectangular sub-grid.

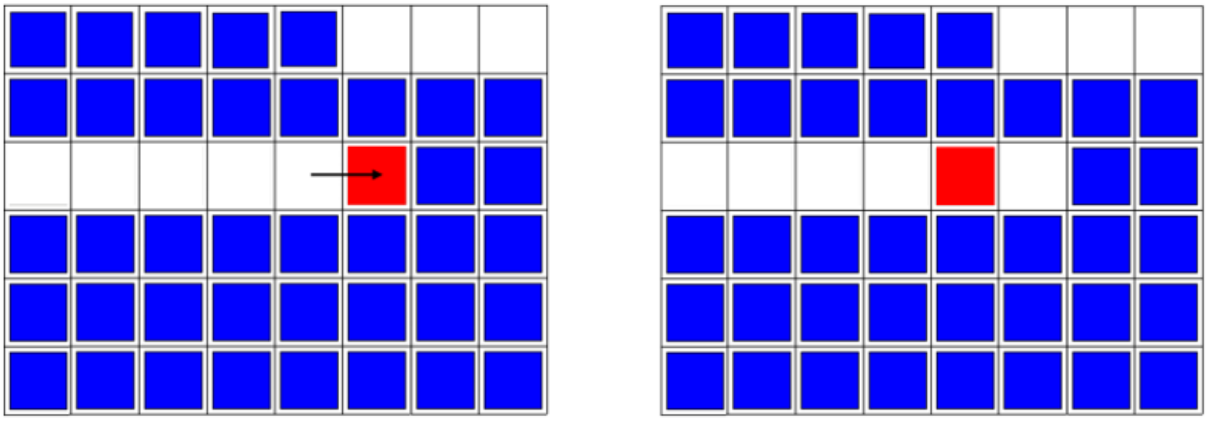
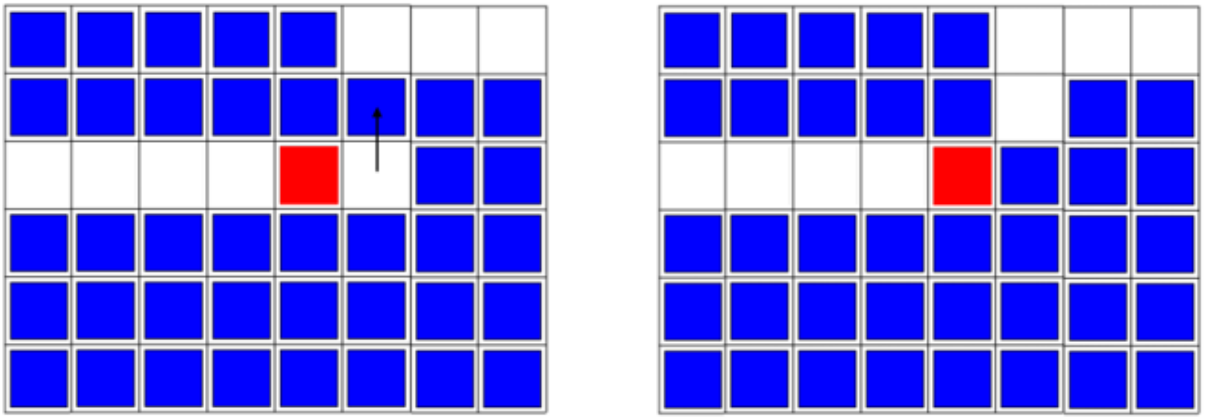
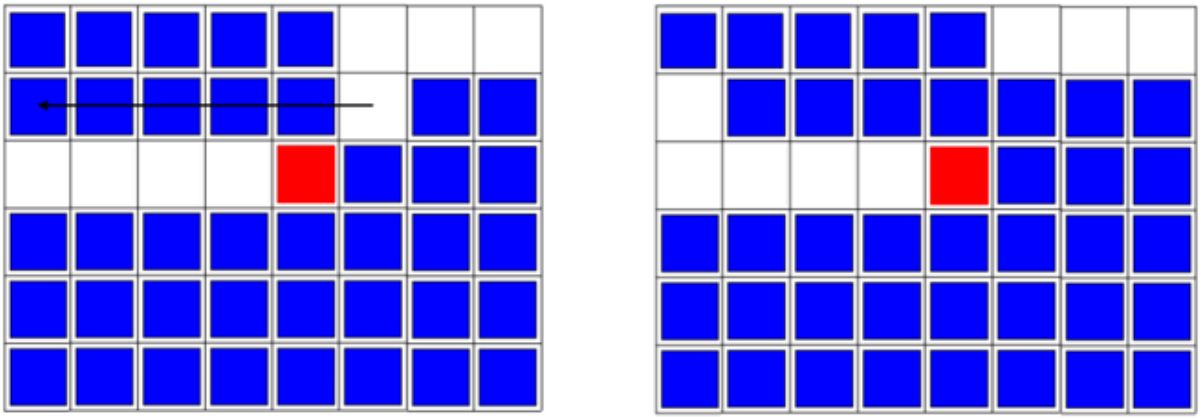


Figure 2.12: RLM performed in a rectangular sub-grid

Finally, only two of the CMs are required and they are responsible for maintaining one escort per column so that a rectilinear aisle can always be formed in one step.
The first CM consists of a single move from the initial cell of the requested load to the adjacent cell in the same column. Note that the initial cell will be occupied by an escort when the first CM is performed. The second CM consists of a block move of the same escort to the cell in the same row adjacent to the square sub-grid. Figure 2.13 illustrates both CMs required.



(a) First CM



(b) Second CM

Figure 2.13: CMs performed in a rectangular sub-grid

Figure 2.14 shows the sequence of all the movements performed in the rectangular sub-grid for the retrieval of a generic requested load.

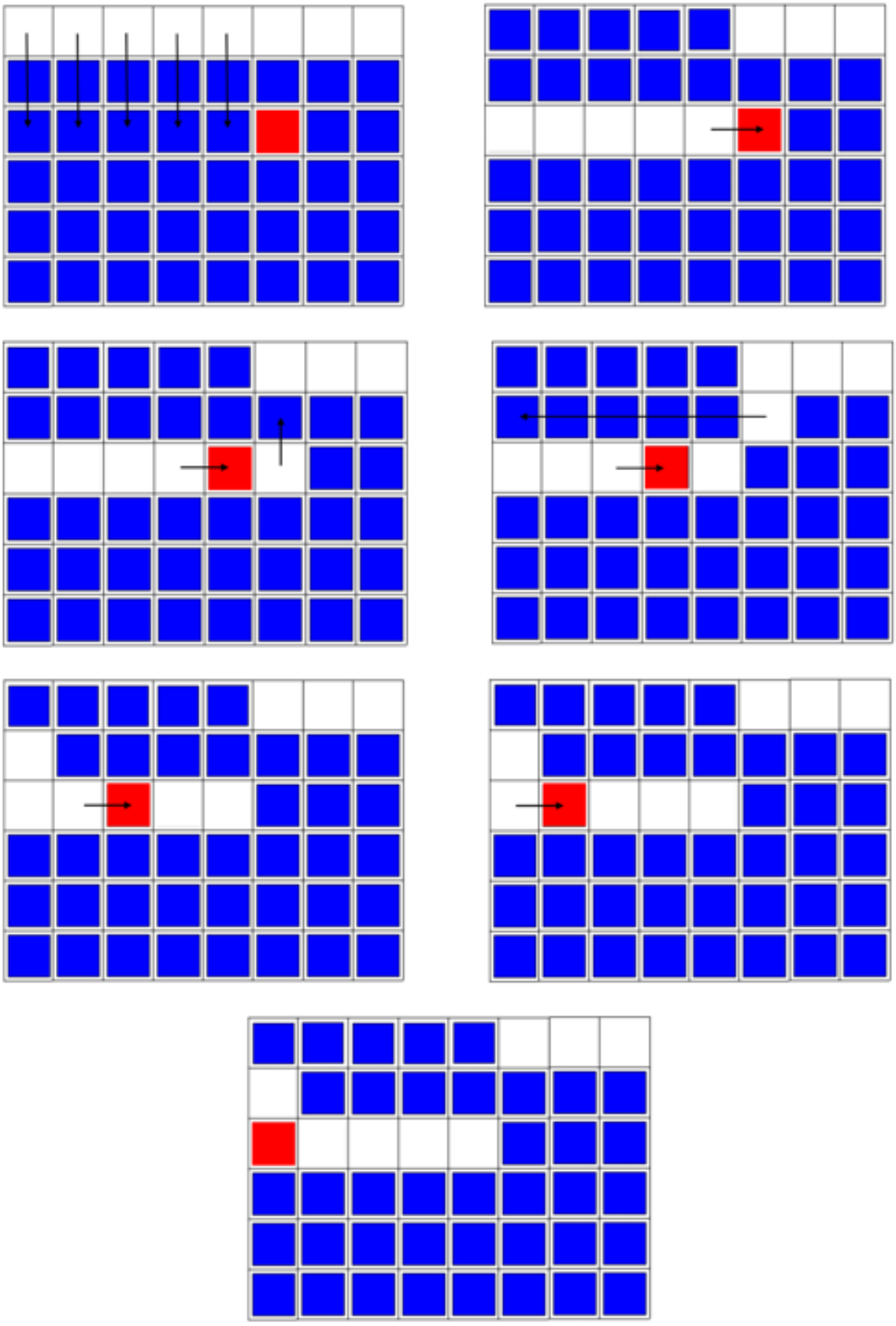


Figure 2.14: Movements required in a rectangular sub-grid

The opening of the rectilinear aisle in the square sub-grid is done when the requested load has almost reached the square sub-grid. An additional movement is required to maintain the number of loads in the sub-grids constant, but it can be performed simultaneously with the final RLM required in the rectangular sub-grid. These final considerations are shown in Figure 2.15.

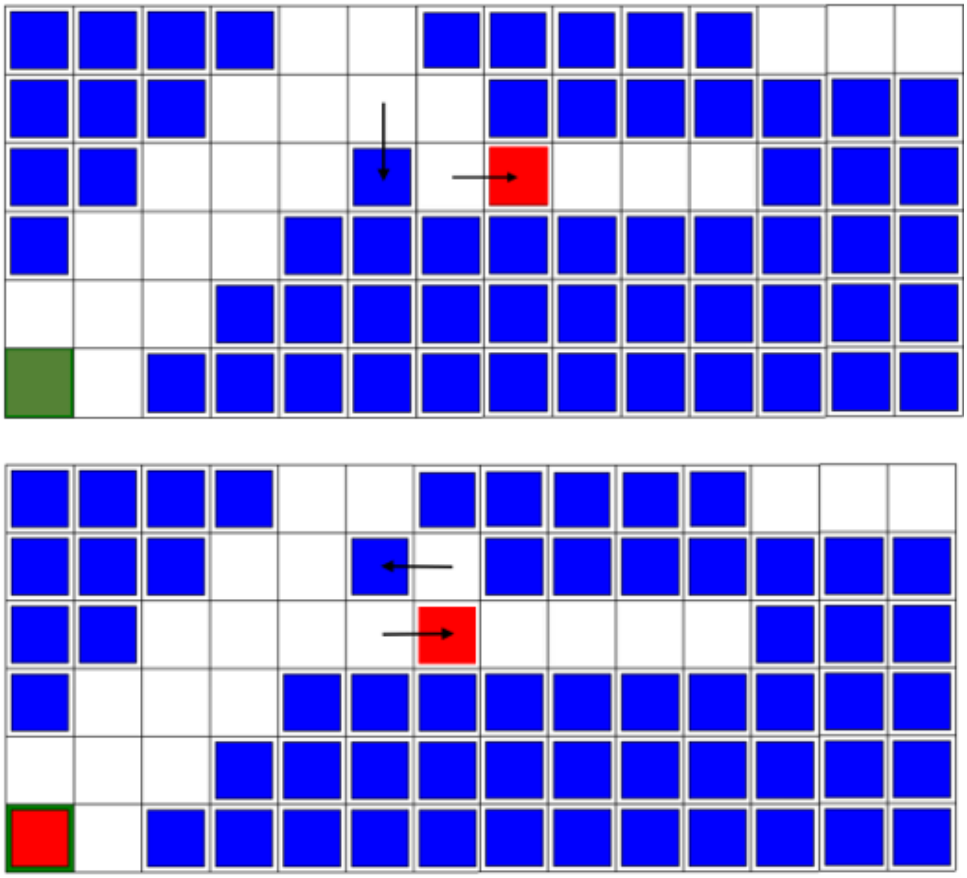


Figure 2.15: Movements required in a rectangular sub-grid

# 3 | Analytical model with random storage policy

This chapter introduces the analytical model for describing the system when the random storage policy is implemented. Sections 3.1 and 3.2 focus on measuring the storage capacity and system density. Section 3.3 provides insights into calculating the average distance traveled by the storage racks. Moving forward, section 3.4 presents the formulas used to determine the average cycle time and throughput. Finally, section 3.5 offers a method for computing the value of $\beta$. This value considers the requirement of opening rectilinear aisles for certain loads.

## 3.1. Storage capacity and density for a square grid

The number of escorts needed to make a diagonal aisle in a square grid of dimensions $m \times m$ is:

$$e\,(m,m) = 3m - 2 \tag{3.1}$$

Consequently, the capacity is the total number of cells in the square grid minus the number of escorts needed to form the aisle:

$$c\,(m,m) = m^2 - (3m-2) = m^2 - 3m + 2 \tag{3.2}$$

The density for a generic square grid is modeled as the capacity divided by the grid size, which is the total number of cells in the grid:

$$\rho\,(m,m) = \frac{m^2 - 3m + 2}{m^2} \tag{3.3}$$

## 3.2. Storage capacity and density for a rectangular grid

In the case of a rectangular grid, the division presented in Section 2.2 is taken into consideration. Hence, the overall number of escorts needed to move a generic load is the sum of the escorts required to form the diagonal aisle and the ones necessary to open the rectilinear aisle.
Considering the rectangular sub-grid of dimensions $m \times n$, a number of escorts equal to $n - m$ is needed to make the rectilinear aisle. Therefore, the overall number of escorts allowing a requested load to travel along the fishbone route in a rectangular grid is:

$$e\,(m,n) = 3m - 2 + (n - m) = 2m + n - 2 \tag{3.4}$$

Consequently, the capacity and the density can be respectively estimated as:

$$c\,(m,n) = mn - 2m + n - 2 = (m-1)(n-2) \tag{3.5}$$

$$\rho\,(m,n) = \frac{(m-1)(n-2)}{mn} \tag{3.6}$$

where $mn$ is the total number of cells in the rectangular grid.

## 3.3. Average travel distance

The cells visited by the requested load to travel from its initial location to the I/O point are the ones included in the fishbone route, which is described in Section 2.2.
The length of this path coincides with the Chebyshev distance but is adjusted considering a factor of $\sqrt{2}$ when moving diagonally in a cell. u indicates the length of the sides of each cell in the grid, measured in meters.

The travel distance between a generic cell and the I/O point can be calculated as:

$$\begin{aligned} d\,(i,j) &= [\; max\,(i;j) - min\,(i;j) + \sqrt{2}\; min\,(i;j)\;]\; u = \\ &= [\; max\,(i;j) + (\sqrt{2} - 1)\; min\,(i;j)\;]\; u \end{aligned} \tag{3.7}$$

where $min(i,j)$ is the number of diagonal moves each load travels on the fishbone route to reach the I/O point. Consequently, $max(i;j) - min(i;j)$ is the number of rectilinear

moves along the fishbone route between $(i; j)$ and the I/O location.
The average travel distance for a square grid is the travel distance between all locations that store loads divided by the capacity. No loads are permanently stored in the diagonal aisle. Therefore, the travel distances from all cells in the diagonal aisle are subtracted.

$$\overline{d}\,(m,m) = \frac{1}{c\,(m,m)}\left(\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} d\,(i,j) - \sum_{i=0}^{m-1} d\,(i,i) - 2\sum_{i=1}^{m-1} d\,(i,(i-1))\right) u \tag{3.8}$$

Similarly, it is possible to estimate the average travel distance for a rectangular grid. The only consideration to make is that loads are not stored in the cells along the upper side of the rectangular sub-grid, so their travel distance is subtracted.

$$\begin{aligned}\overline{d}\,(m,n) = \frac{1}{c\,(m,n)}\Bigg(&\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} d\,(i,j) - \sum_{i=0}^{m-1} d\,(i,i) - \\ &- 2\sum_{i=1}^{m-1} d\,(i,(i-1)) - \sum_{i=m}^{n-1} d\,(i,m-1)\Bigg) u\end{aligned} \tag{3.9}$$

## 3.4. Average cycle time and throughput

The cycle time is the time required to move a load from its initial location to the I/O point and to replenish the grid.
Since the initial configuration of the grid is restored while the requested loads travel to the I/O point, the average cycle time could be easily computed by multiplying the average travel distance by the speed of the loads $v$.
However, as already mentioned in section 2.4, some loads require an additional step to open a rectilinear aisle before moving on the fishbone route. Considering $\beta$ as the fraction of loads in the grid requiring this additional step, the distance traveled by the loads to open a rectilinear aisle is on average equal to $\beta$ times the length of a cell.
Hence, the average cycle time, measured in seconds, can be estimated as:

$$\begin{aligned}\overline{t}\,(m,n) &= \overline{d}\,(m,n) \times v + \beta\,(m,n) \times v = \\ &= \left(\overline{d}\,(m,n) + \beta\,(m,n)\right) v\end{aligned} \tag{3.10}$$

The throughput of the grid measures the number of loads retrieved per hour and can be easily computed as:

$$\tau\,(m,n) = \frac{3600\ s}{\overline{t}\,(m,n)} \tag{3.11}$$

## 3.5. Value of $\beta$

In a generic square grid of dimensions $m \times m$, the loads requiring the additional step to open the rectilinear aisle are all the ones not located next to the diagonal aisle. In this case, the number of loads stored next to the diagonal aisle is equal to $2(m-2)$. Therefore, there are $m^2 - 3m + 2 - 2(m-2) = m^2 - 5m + 6$ loads that have to move through the rectilinear aisle.
Considering the formula for the capacity of a square grid 3.2, the value of $\beta$ in a square grid can be computed as:

$$\beta\,(m,m) = \frac{m^2 - 5m + 6}{m^2 - 3m + 2} \tag{3.12}$$

In the case of a rectangular grid of dimensions $m \times n$, it is necessary to take into account the loads stored in the rectangular sub-grid. Within this area, the probability that a rectilinear aisle does not need to be created when a new load is requested is so low that we assume all loads need an additional step to open a rectilinear aisle. Concretely, there are $m(n-m) - (n-m)$ loads in the rectangular sub-grid requiring the additional step to be performed.
Therefore, the value of $\beta$ for a generic rectangular grid can be computed as:

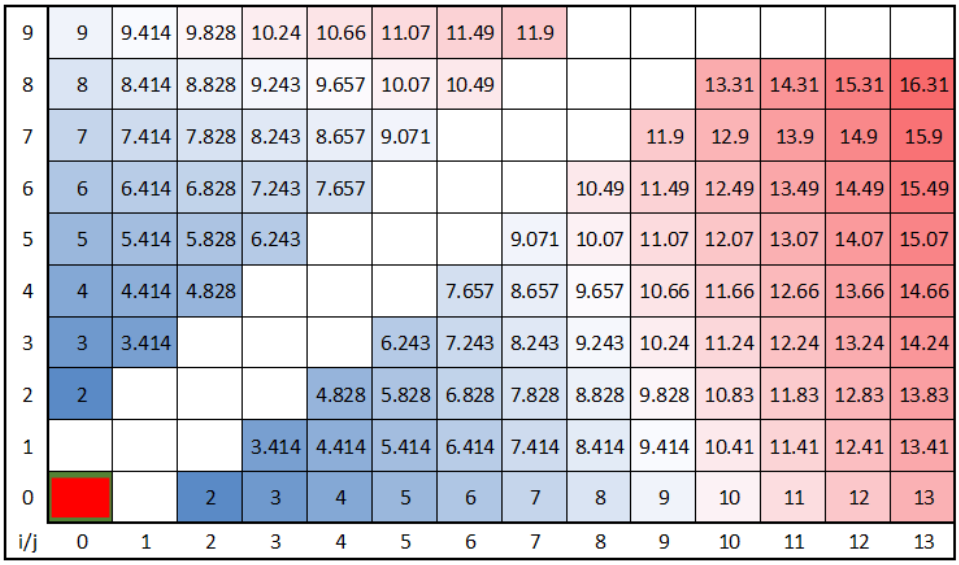
$$\begin{aligned} \beta\,(m,n) &= \frac{m^2 - 5m + 6 + m(n-m) - (n-m)}{(m-1)(n-2)} = \\ &= \frac{mn - 4m - n + 6}{(m-1)(n-2)} \end{aligned} \tag{3.13}$$

# 4 | Analytical model with class-based storage policy

This chapter focuses on the modifications to be made to the analytical model presented in Chapter 3 when implementing the class-based storage policy within the system. It begins by discussing the identification of the shape for class A in Section 4.1 and shows the alternatives that have been selected and analyzed. Section 4.2 demonstrates how the average travel distance, cycle time, and throughput can be computed based on the performance of each class. Section 4.3 presents the analytical model specifically designed for the scenario where class A has a square shape. Finally, in Section 4.4, the focus shifts to the equations and analytical model for the triangular shape of class A.

## 4.1. Classes shape

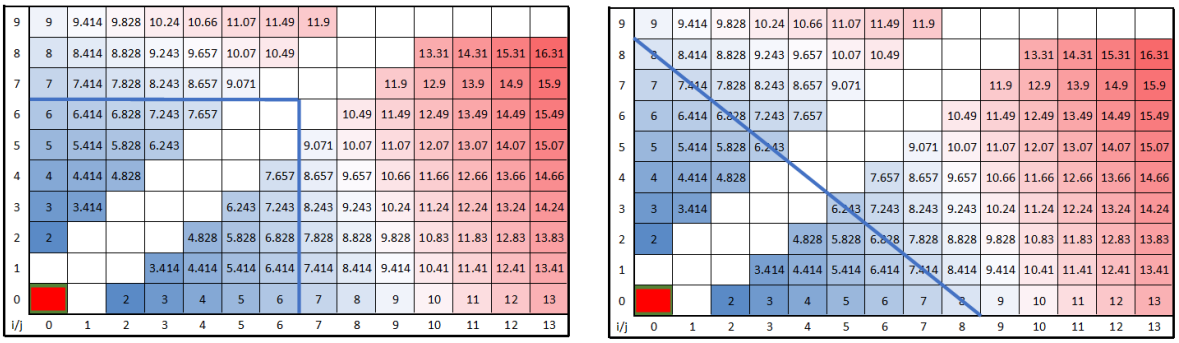
The class-based storage policy with two classes involves categorizing items into class A and class B. The initial phase in designing a class-based storage system is determining the shape of class A. In Figure 4.1, the travel distance required for each load in the grid to reach the I/O point is depicted. The travel distance is represented by a color gradient ranging from blue to red, where the distance increases progressively.



| i/j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 9 | 9.414 | 9.828 | 10.24 | 10.66 | 11.07 | 11.49 | 11.9 | | | | | | |
| 8 | 8 | 8.414 | 8.828 | 9.243 | 9.657 | 10.07 | 10.49 | | | | 13.31 | 14.31 | 15.31 | 16.31 |
| 7 | 7 | 7.414 | 7.828 | 8.243 | 8.657 | 9.071 | | | | 11.9 | 12.9 | 13.9 | 14.9 | 15.9 |
| 6 | 6 | 6.414 | 6.828 | 7.243 | 7.657 | | | | 10.49 | 11.49 | 12.49 | 13.49 | 14.49 | 15.49 |
| 5 | 5 | 5.414 | 5.828 | 6.243 | | | | 9.071 | 10.07 | 11.07 | 12.07 | 13.07 | 14.07 | 15.07 |
| 4 | 4 | 4.414 | 4.828 | | | | 7.657 | 8.657 | 9.657 | 10.66 | 11.66 | 12.66 | 13.66 | 14.66 |
| 3 | 3 | 3.414 | | | | 6.243 | 7.243 | 8.243 | 9.243 | 10.24 | 11.24 | 12.24 | 13.24 | 14.24 |
| 2 | 2 | | | | 4.828 | 5.828 | 6.828 | 7.828 | 8.828 | 9.828 | 10.83 | 11.83 | 12.83 | 13.83 |
| 1 | | | | 3.414 | 4.414 | 5.414 | 6.414 | 7.414 | 8.414 | 9.414 | 10.41 | 11.41 | 12.41 | 13.41 |
| 0 | | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Figure 4.1: Travel distance for each load in the grid

The shape of class A is defined in order to minimize the travel distance for the loads. This work examines two potential shapes: a square shape and a triangular shape. These shapes are illustrated and highlighted in Figure 4.2.



(a) Class A with a square shape

(b) Class A with a triangular shape

Figure 4.2: Possible shapes for class A

## 4.2. Average travel distance, cycle time and throughput

Given the implementation of the random storage policy within each class, the formulas presented in Chapter 3 can be directly applied to each individual class. Consequently, the estimation of the average travel distance and average cycle time for the overall grid

becomes feasible by calculating a weighted average based on the results obtained for each class.
The weights in this weighted average are determined by the picking frequency associated with each class. These picking frequencies are derived by examining the ABC curves, as previously discussed in Section 1.4.

$$\overline{d} = p_A \times \overline{d}_A + p_B \times \overline{d}_B \tag{4.1}$$

$$\bar{t} = p_A \times \bar{t}_A + p_B \times \bar{t}_B \tag{4.2}$$

where $p_B$ can be easily computed as $1 - p_A$.
The throughput of the grid can be computed using the same methodology as previously demonstrated.

$$\tau = \frac{3600\ s}{\bar{t}} \tag{4.3}$$

## 4.3. Class A with square shape

### 4.3.1. Class A for a square grid

Formula 3.2, which applies to a generic square grid, can be utilized to determine the capacity of the class A square sub-grid. By applying this formula, the capacity of the class A sub-grid can be expressed as $a^2 - 3a + 2$.
Assuming that $l_A$ represents the proportion of total loads dedicated to class A items in the grid, it is possible to determine the length of the side of the class A sub-grid. This can be achieved by solving the equation $a^2 - 3a + 2 = l_A(m^2 - 3m + 2)$ with respect to $a$, where $m$ represents the total number of loads. Once the solution is obtained, it is important to round up the result to ensure an integer number of loads.

$$a\ (m, m) = \left\lceil \frac{3 + \sqrt{9 - 4 \times (2 - l_A * (m^2 - 3m + 2))}}{2} \right\rceil \tag{4.4}$$

In Figure 4.3, the shape of class A within a square grid is depicted and highlighted. It illustrates how class A occupies a specific portion of the grid, while the remaining area is dedicated to class B items.

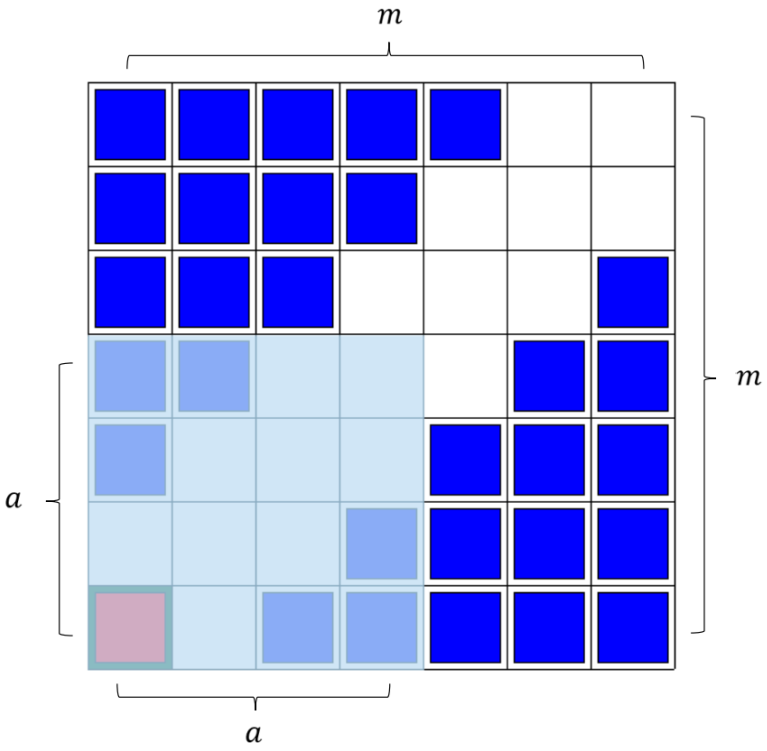



Figure 4.3: Class A with square shape in a square grid

### 4.3.2. Class A for a rectangular grid

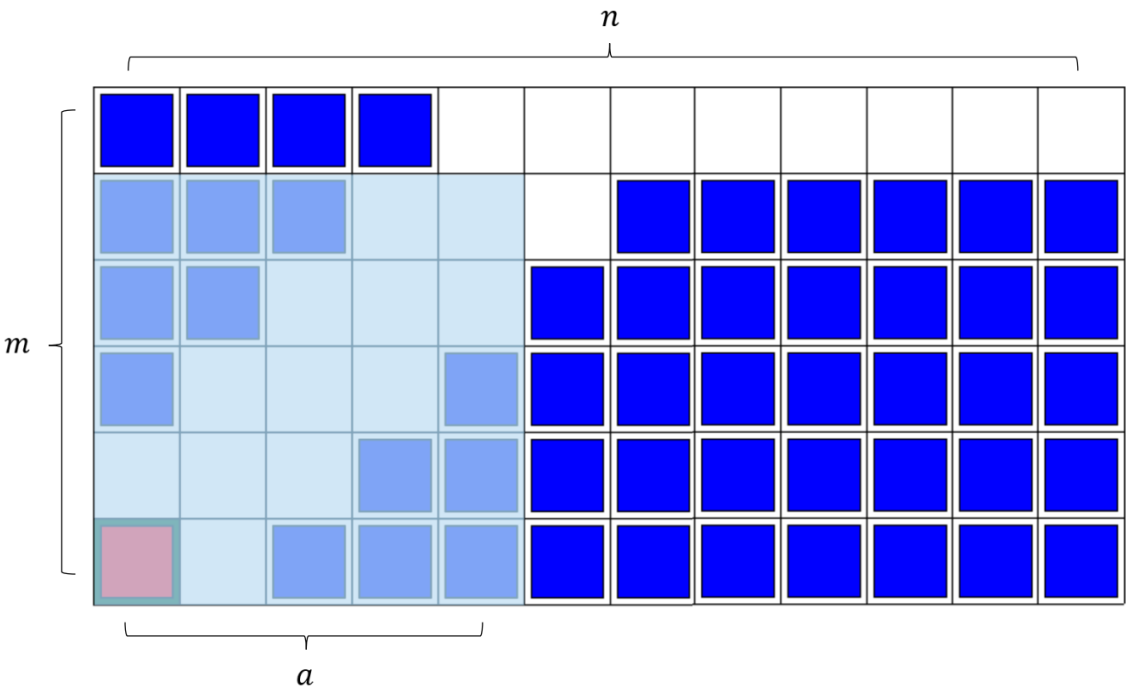
Initially, class A is assumed to have a square shape, similar to the previous case. By examining formula 3.5 for the capacity of a rectangular grid and employing a similar approach, it becomes possible to determine the length of the side for the class A sub-grid. This can be achieved by solving the equation $a^2 - 3a + 2 = l_A(m-1)(n-2)$, where $m$ and $n$ represent the dimensions of the rectangular grid. Following the same procedure, the solution is rounded up to ensure an integer number of loads.

$$a\ (m,n) = \left\lceil \frac{3 + \sqrt{9 - 4 \times [2 - l_A * (m-1)(n-2)]}}{2} \right\rceil \tag{4.5}$$

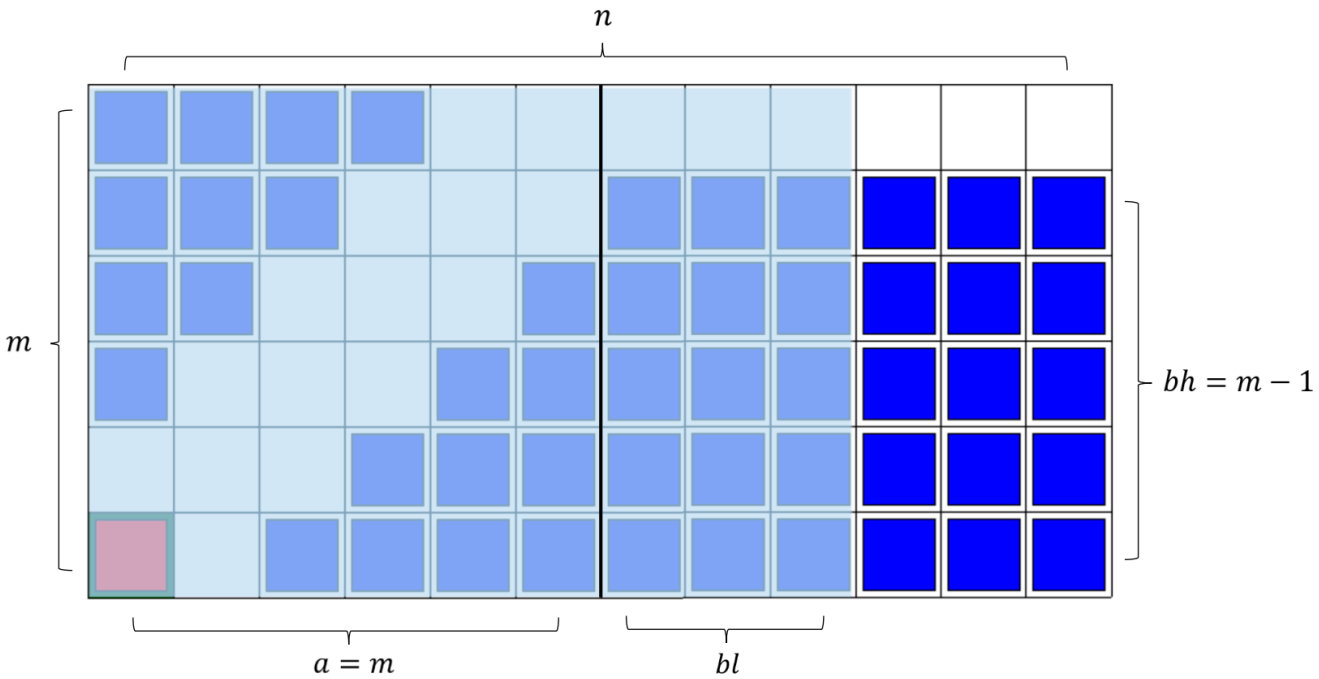
Based on the obtained result, it is important to differentiate between the following scenarios:

- If $a \leq m$, the class A is a square sub-grid with dimension $a \times a$
- If $a > m$, class A is composed of a square sub-grid of dimension $m \times m$, plus an additional area of dimensions $bl \times bh$

Figure 4.4 shows the possible shapes of class A in a rectangular grid.




(a) $a \leq m$




(b) $a > m$

Figure 4.4: Class A with square shape in a rectangular grid

## Average travel distance and cycle time for $a \leq m$

Since Class A is a square sub-grid with dimensions $a \times a$, the formulas previously presented for a generic square grid can indeed be applied to estimate its capacity, density, and the value of $\beta$.

$$c_A = a^2 - 3a + 2 \tag{4.6}$$

$$\rho_A = \frac{a^2 - 3a + 2}{a^2} \tag{4.7}$$

$$\beta_A = \frac{a^2 - 5a + 6}{a^2 - 3a + 2} \tag{4.8}$$

As previously described, the average travel distance within Class A can be estimated by summing the travel distance for each cell within this area that stores a load, and then dividing it by the capacity of Class A. Specifically, formula 3.8 can be used for this purpose, with adjustments made to account for the specific capacity and dimensions of the Class A sub-grid.

$$\overline{d}_A = \frac{1}{c_A}\left(\sum_{i=0}^{a-1}\sum_{j=0}^{a-1} d\,(i,j) - \sum_{i=0}^{a-1} d\,(i,i) - 2\sum_{i=1}^{a-1} d\,(i,(i-1))\right) u \tag{4.9}$$

The average travel distance for class B can be devised as:

$$\overline{d}_B = \frac{\overline{d}_A \times c_A + \overline{d}_{grid} \times c_{grid}}{c_B} \tag{4.10}$$

where $\overline{d}_{grid}$ differs depending on the shape of the overall grid and it is computed using either formula 3.8 or 3.9.

The shape of class B becomes relevant in estimating the value of $\beta$ and computing the average cycle time. In the case of a rectangular grid where $a = m$, the shape of class B is straightforward. It corresponds to the rectangular sub-grid with dimensions $m \times (m - n)$, and the number of loads requiring a rectilinear aisle to be opened can be calculated as $m(n - m) - (n - m)$, as previously estimated.
However, in all other cases where $a \neq m$, the shape of class B is more complex. Therefore, the value of $\beta_B$ is estimated indirectly by using the grid and class A as reference points. Specifically, the number of loads in class B that require an additional step is computed by subtracting the number of loads in class A requiring this additional move from the total

number of loads requiring it in the grid. Similarly, the capacity of class B is obtained as the difference between the overall grid capacity and the capacity of class A.
In the specific case of a square grid, the final result is:

$$\beta_B = \frac{m^2 - 5m + 6 - (a^2 - 5a + 6)}{m^2 - 3m + 2 - (a^2 - 3a + 2)} \tag{4.11}$$

The one for a rectangular grid is:

$$\beta_B = \frac{mn - 4m - n + 6 - (a^2 - 5a + 6)}{(m-1)(n-2) - (a^2 - 3a + 2)} \tag{4.12}$$

Finally, the average cycle time for each class is computed using the formula 3.10.

### Average travel distance and cycle time for $a > m$

From the right image in Figure 4.4, it is evident that when $a > m$, class A takes on a rectangular shape comprising both a square sub-grid with dimensions $m \times m$ and a rectangular sub-grid with dimensions $bl \times bh$.
The total number of loads within class A is intended to be $l_A(m-1)(n-2)$. The capacity of the square sub-grid is $m^2-3m+2$. Thus, the rectangular sub-grid should accommodate a number of loads equal to $l_A(m-1)(n-2)-(m^2-3m+2)$. Consequently, the length of this rectangular sub-grid can be determined by solving the equation $l_A(m-1)(n-2) - (m^2 - 3m + 2) = bl \times bh$, taking into account that $bh = m - 1$.

$$bl = \left\lceil \frac{l_A(m-1)(n-2) - (m^2 - 3m + 2)}{m-1} \right\rceil \tag{4.13}$$

The capacity and density for class A in this case are:

$$c_A = m^2 - 3m + 2 + (bl \times (m-1)) \tag{4.14}$$

$$\rho_A = \frac{m^2 - 3m + 2 + (bl \times (m-1))}{m \times (m + bl)} \tag{4.15}$$

The assumption already made about the number of loads in a rectangular sub-grid requiring the rectilinear aisle to be opened is still valid. Consequently, the value of $\beta$ for class A is computed as:

$$\beta_A = \frac{m^2 - 5a + 6 + (bl \times (m-1))}{m^2 - 3m + 2 + (bl \times (m-1))} \tag{4.16}$$

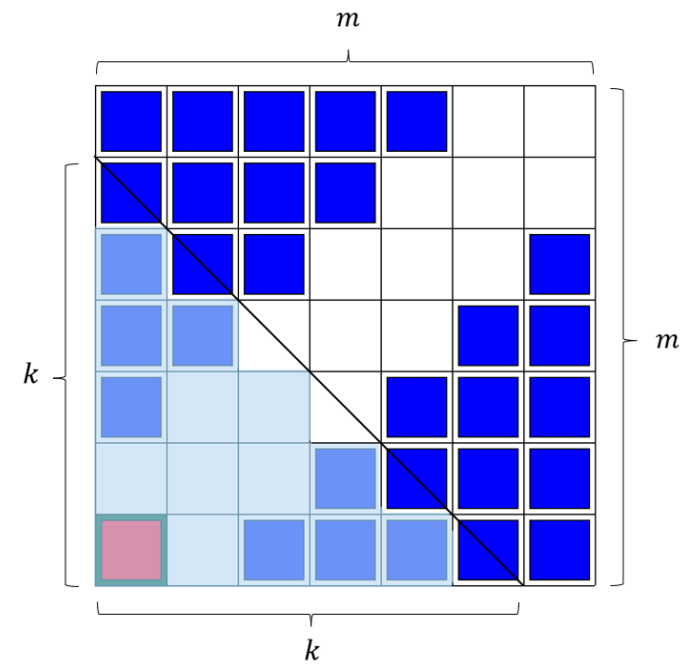
The average travel distance within this class could be easily devised starting from formula 3.9 and adjusting the capacity and the length of the grid:

$$\begin{aligned}\overline{d}_A = \frac{1}{c_A}\Bigg( \sum_{i=0}^{m-1} \sum_{j=0}^{(m+bl)-1} d\,(i,j) - \sum_{i=0}^{m-1} d\,(i,i) - \\ - 2\sum_{i=1}^{m-1} d\,(i,(i-1)) - \sum_{i=m}^{(m+bl)-1} d\,(i,m-1)\Bigg) u\end{aligned} \tag{4.17}$$

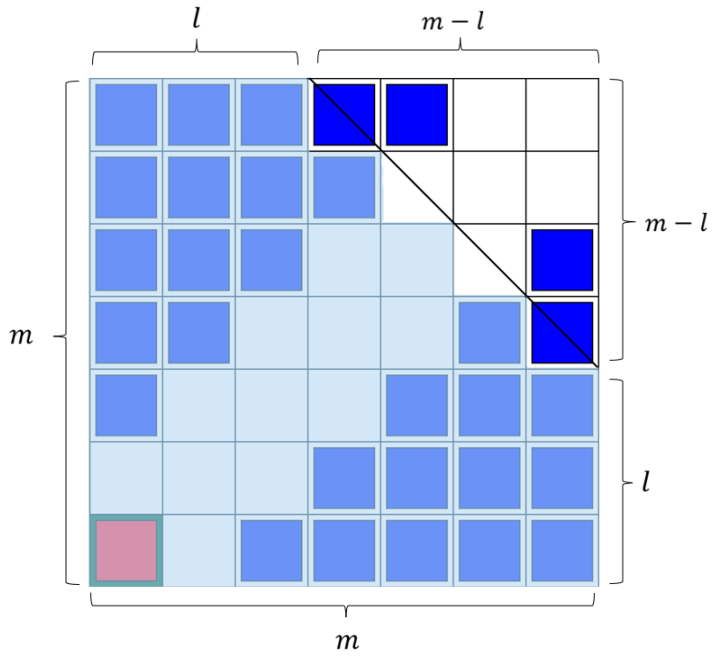
Again, the average travel distance for class B is obtained as in 4.10 and the value of $\beta_B$ is 1 since all the loads stored in this class required a rectilinear aisle to move through, under the assumption made previously. Finally, formula 3.10 is used to estimate the average cycle time within each class.

## 4.4. Class A with triangular shape

### 4.4.1. Class A for a square grid




(a) $k \leq m$




(b) $k > m$

Figure 4.5: Class A with triangular shape in a square grid

### Average travel distance and cycle time for $k \leq m$

Class A is a sub-grid with an isosceles right triangle shape and the length of the sides is equal to $k$. An example of this configuration is shown in Figure 4.5 a. To manage this case, it is necessary to introduce specific formulas, which are different depending on the fact that $k$ is odd or even.
When $k$ is odd, the capacity of class A is estimated as:

$$c_A = 2 \sum_{i=1}^{\frac{k-1}{2}-1} 2i \tag{4.18}$$

When $k$ is even, instead, the capacity is equal to:

$$c_A = 2 \sum_{i=1}^{\frac{k}{2}-1} 2i - 1 \tag{4.19}$$

These equations can be used within an iterative process aimed at finding the minimum value of $k$ allowing the class A to store $l_A$ of the total number of loads in the grid. This iterative process can be described in the following way:

1. Set $k = 4$, which is the minimum value in order to have loads in the sub-grid;
2. Compute the capacity associated with the current value of k, using the formulas already presented;
3. Compute the number of loads that should be stored in class A as $l_A(m^2 - 3m + 2)$
4. If the actual capacity is higher or equal to the minimum capacity required for class A, the actual value of k is the optimal one and the process ends. If this is not the case, increase by 1 the value of k and iterate the process from 2.

If a solution is not found with this iterative process, it means that a larger class A shape is required. See the case of $k > m$ explained in the next paragraph.

Once identified the value of $k$ and the respective capacity for class A, it is possible to proceed to compute the total number of cells ($N_A$), both storing loads and empty, within this area:

$$N_A = \sum_{i=1}^{k-1} i \tag{4.20}$$

Then the density can be easily computed as the ratio between the capacity and the total

number of cells in the sub-grid.

$$\rho_A = \frac{c_A}{N_A} \tag{4.21}$$

Finally, the last parameter to be estimated is $\beta$ and its computation differs in the case of $k$ odd or even. In general, it is found by subtracting from the number of loads stored in class A, the ones close to the diagonal, which do not require the rectilinear aisle to be opened. This value is then subdivided by the class A capacity.
When $k$ is odd, it can be computed in the following way:

$$\beta_A = \frac{c_A - 2\left(\frac{k-1}{2} - 1\right)}{c_A} \tag{4.22}$$

When $k$ is even, the formula is slightly different

$$\beta_A = \frac{c_A - 2\left(\frac{k}{2} - 1\right)}{c_A} \tag{4.23}$$

For computing the average travel distance within class A, the limits of the summation terms in Formula are adjusted and the result for $k$ odd is given by:

$$\overline{d}_A = \frac{1}{c_A}\left(\sum_{i=0}^{k-2}\sum_{j=0}^{k-2-i} d\,(i,j) - \sum_{i=0}^{\frac{k-1}{2}-1} d\,(i,i) - 2\sum_{i=1}^{\frac{k-1}{2}} d\,(i,(i-1))\right)u \tag{4.24}$$

In the case of $k$ even, the average travel distance can be estimated as:

$$\overline{d}_A = \frac{1}{c_A}\left(\sum_{i=0}^{k-2}\sum_{j=0}^{k-2-i} d\,(i,j) - \sum_{i=0}^{\frac{k}{2}-1} d\,(i,i) - 2\sum_{i=1}^{\frac{k}{2}-1} d\,(i,(i-1))\right)u \tag{4.25}$$

Similarly to what has been already explained in the previous chapter, the formulas for class B are retrieved as a difference with respect to the overall grid, and formulas 4.2 and 4.3 can be used to measure the average cycle time and the throughput.
The estimation of the value of $\beta$ for class B is slightly more complex than that of the other parameters. It is computed starting from the total capacity of class B and subtracting the difference between the total number of loads close to the diagonal $(m-2)$ and the number of loads close to the diagonal and belonging to class A. In formulas, when $k$ is odd:

$$\beta_B = \frac{c_B - 2\left(m - 2 - \left(\frac{k-1}{2} - 1\right)\right)}{c_B} \tag{4.26}$$

When $k$ is even:

$$\beta_B = \frac{c_B - 2\left(m - 2 - \left(\frac{k}{2} - 1\right)\right)}{c_B} \tag{4.27}$$

## Average travel distance and cycle time for $k > m$

Figure 4.5 b shows the shape of the class A sub-grid in the case of $k > m$. The shape is no more triangular but pentagonal. With a similar procedure as the one used to find $k$, it is possible to find the length of $l$, which is the only unknown dimension.
With the aim of facilitating the analysis of this case, the class A sub-grid is subdivided into 2 smaller areas, as shown in Figure 4.6.

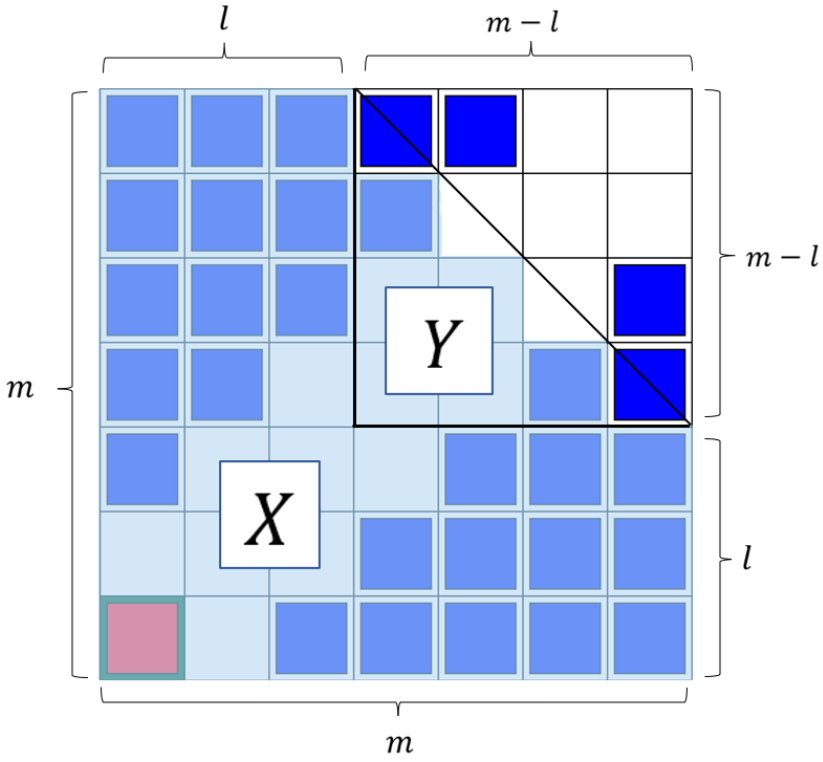



Figure 4.6: Subdivision of class A for $k > m$

Thanks to this subdivision, it is possible to implement for $Y$ the formulas already presented for the case $k \leq m$, by adjusting the value of $k$, which coincides with $m - l$.
Retrieving formulas for $X$ is also intuitive. The capacity for this area is simply given by:

$$c_X = 2\sum_{i=0}^{l-1}(m - 2 - i) \tag{4.28}$$

Therefore, the total capacity of class A is obtained as the sum of the capacity of $X$ and

the capacity of $Y$. The result when $m-l$ is odd is equal to:

$$c_A = 2\sum_{i=0}^{l-1}(m-2-i) + 2\sum_{i=1}^{\frac{m-l-1}{2}-1} 2i \tag{4.29}$$

In the case of $m-l$ even, it is given by:

$$c_A = 2\sum_{i=0}^{l-1}(m-2-i) + 2\sum_{i=1}^{\frac{m-l}{2}-1} 2i-1 \tag{4.30}$$

The same procedure explained in the previous paragraph is applied with the aim of finding the minimum value of $l$ in order to have the number of loads required in class A. The only difference is that the starting value for $l$ is equal to 1. Once identified the optimal value, the associated value of the capacity is used to continue with the analysis.
The total number of cells in the area dedicated to class A is:

$$N_A = ml + (m-l)\,l + \sum_{i=1}^{m-l-1} i \tag{4.31}$$

The equations for the density are not reported but they are simply obtained by dividing the capacity by the total number of cells.
The estimation of the value of $\beta$ is slightly more complex, thus it is again better to look at the subareas separately. To facilitate understanding, the variable $\lambda$ is introduced and it measures the number of loads requiring the additional step. This value is computed for both subareas. Note that the value for $Y$ is again different depending on whether $m-l$ is odd or even.

$$\lambda_X = 2\sum_{i=0}^{l-1}(m-3-i) \tag{4.32}$$

$$\lambda_Y = 2\left(\sum_{i=1}^{\frac{m-l-1}{2}-1} 2i\right) - 2\left(\frac{m-l-1}{2}-1\right) \tag{4.33}$$

If $m-l$ is odd, otherwise:

$$\lambda_Y = 2\left(\sum_{i=1}^{\frac{m-l}{2}-1} 2i-1\right) - 2\left(\frac{m-l}{2}-1\right) \tag{4.34}$$

Starting from these results, it is easier to compute $\beta$. It is sufficient to sum these values

and divide the result by the total capacity of class A.

$$\beta_A = \frac{\lambda_X + \lambda_Y}{c_A} \tag{4.35}$$

The subdivision of the sub-grid is taken into account also for the computation of the average travel distance. Referring to formulas 4.24 and 4.25, it is easy to compute the total travel distance within area Y. It is computed in the following ways, respectively for $m - l$ odd and even:

$$d_Y = \left( \sum_{i=l}^{m-2} \sum_{j=l}^{m-2-(i-l)} d\,(i,j) - \sum_{i=l}^{l+\frac{m-l-1}{2}-1} d\,(i,i) \; - 2 \sum_{i=l+1}^{l+\frac{m-l-1}{2}} d\,(i,(i-1)) \right) u \tag{4.36}$$

$$d_Y = \left( \sum_{i=l}^{m-2} \sum_{j=l}^{m-2-(i-l)} d\,(i,j) - \sum_{i=l}^{l+\frac{m-l}{2}-1} d\,(i,i) \; - 2 \sum_{i=l+1}^{l+\frac{m-l}{2}-1} d\,(i,(i-1)) \right) u \tag{4.37}$$

The total travel distance in X is easily computed as:

$$d_X = \left( 2 \sum_{i=0}^{l-1} \sum_{j=2+i}^{m-1} d\,(i,j) \right) u \tag{4.38}$$

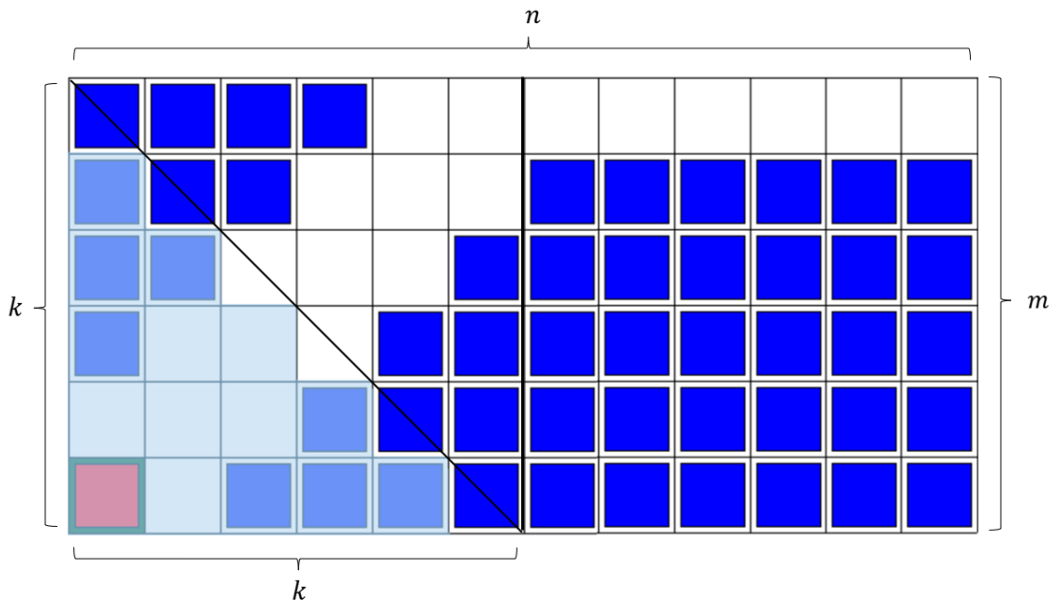
The average travel distance can be obtained as:

$$\overline{d}_A = \frac{d_Y + d_X}{c_A} \tag{4.39}$$

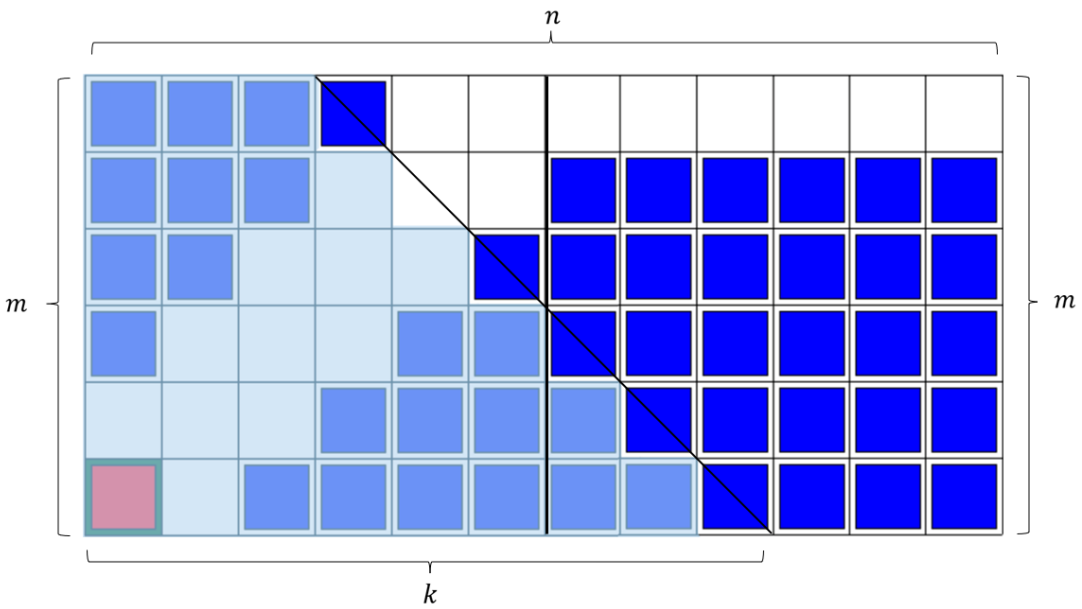
Finally, the shape of class B is the same as the one of class A in the case $k \leq m$. Therefore, for the estimation of the value of $\beta$ for class B, it is sufficient to use the formula and adjust $k$ with $m - l + 1$ and differentiate depending on whether $m - l + 1$ is odd or even.
For both cases, the average cycle time and throughput for each area can be estimated once again by applying formulas 3.10 and 3.11. Then, the final results can be obtained by referring to formulas 4.2 and 4.3.

### 4.4.2. Class A for a rectangular grid

The possible configurations for class A in a rectangular grid vary depending on the shape ratio of the grid. However, if we assume that the area dedicated to class A is significantly smaller than the overall grid (between 10% and 35%), as is commonly designed in practice, the possible configurations can be summarized in the cases shown in Figure 4.7.




(a) $k \leq m$




(b) $k > m$

Figure 4.7: Class A with triangular shape in a rectangular grid

## Average travel distance and cycle time for $k \leq m$

As in the same case already presented for a square grid, class A is a sub-grid with an isosceles right triangle shape and the length of the sides is equal to $k$. For this reason, the estimation of all the parameters related to class A can be obtained using the same formulas.

Notice that, to find the minimum value of k that allows having the required number of loads in class A, it is necessary to change the number of loads required to $l_A(m-1)(n-2)$.

Different considerations can be done for class B, whose shape is totally different from the previous case. However, the capacity and the average travel distance for this class can be retrieved taking into account the overall grid and the class A sub-grid once again.

The only parameter remaining to be estimated is $\beta$. Again, for simplicity class B is subdivided. For the rectangular sub-grid, it is assumed that all the loads require the rectilinear aisle to be opened and the capacity of this area is $m(n-m)$. The number of loads requiring the additional step in the other area can be computed starting from equations 4.33 and 4.34 and adjusting the value of $l$, which in our case is $m-k$.

Overall, the value of $\beta$ for class B can be estimated as:

$$\beta_B = \frac{m(n-m) + 2\left(\sum_{i=1}^{\frac{m-(m-k)-1}{2}-1}(2i) - 2\left(\frac{m-(m-k)-1}{2}-1\right)\right)}{c_B} \tag{4.40}$$

when m-k is odd, otherwise as:

$$\beta_B = \frac{m(n-m) + 2\left(\sum_{i=1}^{\frac{m-l}{2}-1}(2i-1) - 2\left(\frac{m-l}{2}-1\right)\right)}{c_B} \tag{4.41}$$

### Average travel distance and cycle time for $k > m$

In this configuration, class B has the shape of a trapezoid whose base is equal to $k$ and height equal to $m$.
The results for class A are obtained by extending its area in order to have a triangle with both sides equal to $k$, and then subtracting the area that is not included in the original shape. This idea is shown in Figure 4.8.

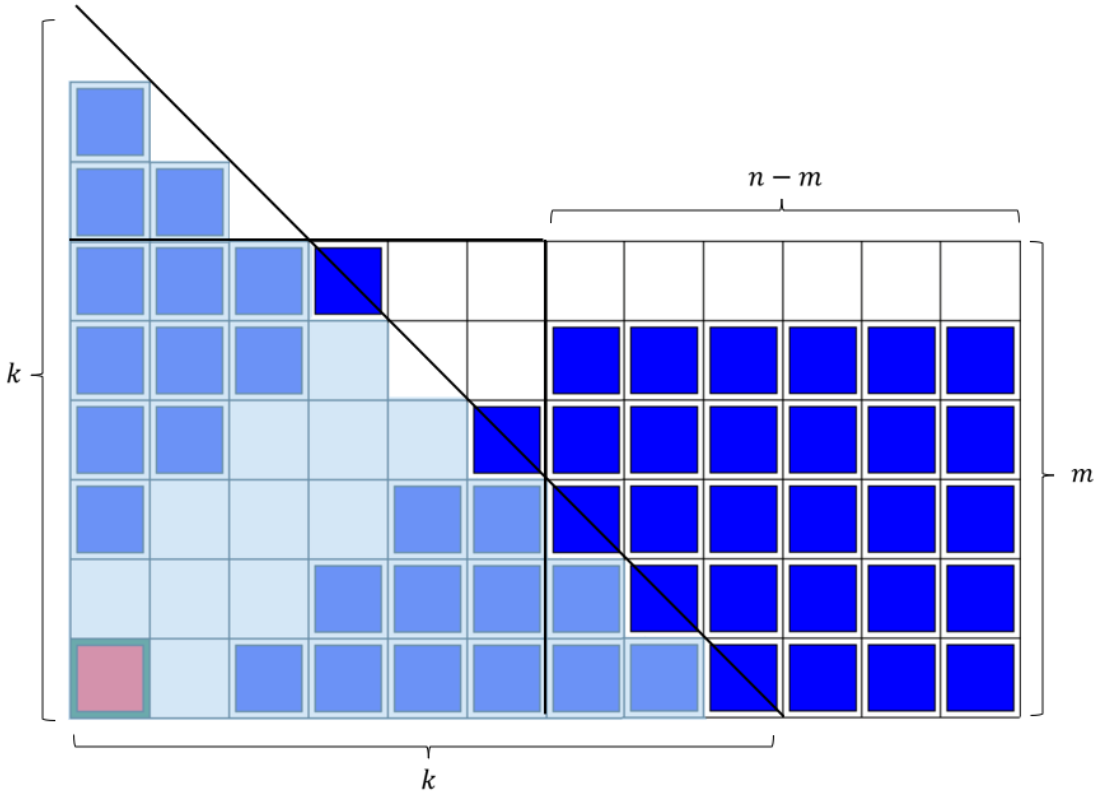



Figure 4.8: Subdivision of class A for $k > m$

Following this approach, it is possible to estimate the parameters starting from the formulas presented for the case $k \leq m$ in a square grid. The main issue consists in adjusting them to remove the additional loads that have been introduced.
When $k$ is odd, the capacity can be estimated as:

$$c_A = 2 \sum_{i=1}^{\frac{k-1}{2}-1} (2i) - \sum_{i=1}^{k-m-1} i \tag{4.42}$$

When $k$ is even, instead, the capacity is equal to:

$$c_A = 2\sum_{i=1}^{\frac{k}{2}-1}(2i-1) - \sum_{i=1}^{k-m-1} i \tag{4.43}$$

These formulas are used to find the actual class A dimension and its capacity, following always the same procedure.
Similarly, for the total number of cells ($N_A$) within class A:

$$N_A = \sum_{i=1}^{k-1} i - \sum_{i=1}^{k-m-1} i \tag{4.44}$$

For the average travel distance, again, it is sufficient to refer to the formulas for a square grid and $k \leq m$ and adjust them by removing the distance traveled by the additional loads.
When $k$ is odd, the average travel distance within class A is:

$$\overline{d}_A = \frac{1}{c_A}\Bigg( \sum_{i=0}^{k-2}\sum_{j=0}^{k-2-i} d\,(i,j) - \sum_{i=0}^{\frac{k-1}{2}-1} d\,(i,i) \; - 2\sum_{i=1}^{\frac{k-1}{2}} d\,(i,(i-1)) - \sum_{j=0}^{k-m-2}\sum_{i=m}^{k-j-2} d\,(i,j)\Bigg) u \tag{4.45}$$

In the case of $k$ even, the average travel distance can be estimated as:

$$\overline{d}_A = \frac{1}{c_A}\Bigg( \sum_{i=0}^{k-2}\sum_{j=0}^{k-2-i} d\,(i,j) - \sum_{i=0}^{\frac{k}{2}-1} d\,(i,i) \; - 2\sum_{i=1}^{\frac{k}{2}-1} d\,(i,(i-1)) - \sum_{j=0}^{k-m-2}\sum_{i=m}^{k-j-2} d\,(i,j)\Bigg) u \tag{4.46}$$

The estimation of the value of $\beta$ for this class is straightforward. The results for $k$ odd and even are respectively:

$$\beta_A = \frac{c_A \; - 2\left(\frac{k-1}{2}-1\right)}{c_A} \tag{4.47}$$

$$\beta_A = \frac{c_A \; - 2\left(\frac{k}{2}-1\right)}{c_A} \tag{4.48}$$

This parameter must be estimated also for class B and for this purpose the rectangular sub-grid is again subdivided. Because of our assumption, the number of loads requiring the rectilinear aisle to be open in the rectangular area is equal to the total number of loads in this area, but then the ones belonging to class A must be subtracted. As regards

the other area, formulas 4.22 and 4.23 can be taken as reference, where $k$ in our case is equal to $m-(k-m)+1$.
From this considerations, the value of $\beta$ for class B can be devised differently depending on whether $m-(k-m)+1$ is odd or even respectively:

$$\beta_B = \frac{2\sum_{i=1}^{\frac{m-(k-m)}{2}-1}(2i) \; - 2\left(\frac{m-(k-m)}{2}-1\right) + \; \left(\; (m-1)(n-m) - \sum_{i=1}^{k-m-1} i \;\right)}{c_B} \tag{4.49}$$

$$\beta_B = \frac{2\sum_{i=1}^{\frac{m-(k-m)+1}{2}-1}(2i-1) \; - 2\left(\frac{m-(k-m)+1}{2}-1\right) + \; \left(\; (m-1)(n-m) - \sum_{i=1}^{k-m-1} i \;\right)}{c_B} \tag{4.50}$$

Finally, the cycle time and the throughput for each class can be easily computed, followed by the overall performance of these configurations.

# 5 | Design procedure

In this section, we will try to define a design procedure. Several input parameters are included in the analysis with the aim to find the optimal configuration for different scenarios. The optimal solution is defined as the configuration allowing to maximize the throughput, which coincides with the one allowing to minimize the average cycle time.
The first input parameter analyzed is the percentage of racks dedicated to class A items ($l_A$). It is a function of the ABC curve and it has a direct consequence on the probability that a requested load belongs to class A, as already explained in section 1.4.
The analysis focuses on the ABC curves 60/20, 70/20, 80/20 and 90/20. For each of these possible values, the optimal percentage of racks dedicated to class A ($l_A{}^*$), which is the one allowing to minimize the average cycle time, depends on the shape ratio of the grid. The shape ratio is another input parameter for the design procedure and it is defined as the ratio between the dimension of the sides of the overall grid:

$$sr = \frac{n}{m} \tag{5.1}$$

The relationship between the optimal value of $l_A$ and the shape ratio of the grid can be analyzed for each ABC curve. These relationships have been derived starting from finding the optimal value of $l_A$ for various grid capacity values, including $4900m^2$, $6400m^2$, $8100m^2$, $10000m^2$, and $121000m^2$.
Figures 5.1 depict the linear regressions fitted to the results obtained using each ABC curve. Each of these regression lines can be represented respectively through the following equations:

$$60/20: \; l_A{}^* = 0.185 + 0.040 \; sr \tag{5.2}$$

$$70/20: \; l_A{}^* = 0.156 + 0.030 \; sr \tag{5.3}$$

$$80/20: \; l_A{}^* = 0.122 + 0.021 \; sr \tag{5.4}$$

$$90/20: \; l_A{}^* = 0.084 + 0.012 \; sr \tag{5.5}$$

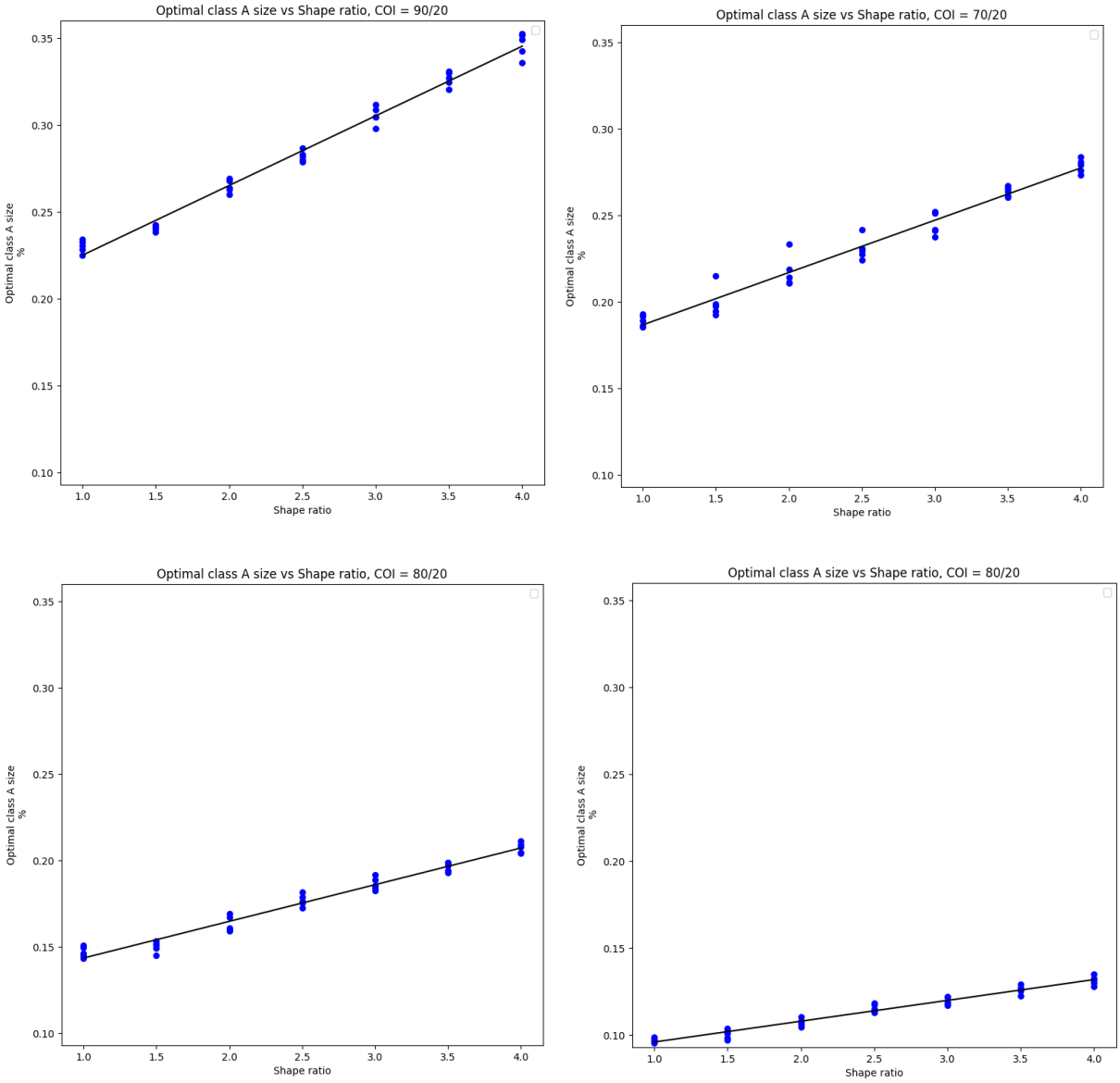



Figure 5.1: Optimal $l_A$ vs shape ratio with different ABC curves and square shape

In the analysis, the shape ratio values considered are 1, 2, 3, and 4. These values are used to compute the corresponding optimal percentages of racks dedicated to class A items using the equations mentioned above. Tables 5.1-5.4 present the optimal $l_A^*$ values, along with the associated probabilities that a requested load belongs to the class A area, for the different ABC curves.

| | *Shape ratio* | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $l_A*$ | 23% | 27% | 31% | 35% |
| $p_A*$ | 64% | 69% | 73% | 76% |

Table 5.1: Optimal $l_A*$ and relative $p_A*$ with ABC curve 60/20

| | *Shape ratio* | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $l_A*$ | 19% | 22% | 25% | 28% |
| $p_A*$ | 69% | 73% | 76% | 78% |

Table 5.2: Optimal $l_A*$ and relative $p_A*$ with ABC curve 70/20

| | *Shape ratio* | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $l_A*$ | 14% | 16% | 19% | 21% |
| $p_A*$ | 72% | 75% | 79% | 81% |

Table 5.3: Optimal $l_A*$ and relative $p_A*$ with ABC curve 80/20

| | *Shape ratio* | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $l_A*$ | 10% | 11% | 12% | 13% |
| $p_A*$ | 80% | 82% | 83% | 84% |

Table 5.4: Optimal $l_A*$ and relative $p_A*$ with ABC curve 90/20

The same procedure can be applied when considering a triangular shape for class A. Similar results can be obtained, and regression lines can be fitted to represent the relationship between the optimal values and the shape ratios for the different ABC curves.

The following equations represent the regression lines fitted on the results:

$$60/20: \; {l_A}^* = 0.186 + 0.041 \; sr \tag{5.6}$$

$$70/20: \; {l_A}^* = 0.156 + 0.030 \; sr \tag{5.7}$$

$$80/20: \; {l_A}^* = 0.123 + 0.021 \; sr \tag{5.8}$$

$$90/20: \; {l_A}^* = 0.083 + 0.013 \; sr \tag{5.9}$$

Figure 5.2 shows the respective regression lines. The optimal values of $l_A$ estimated through the equations are the same already obtained with a class A with a square shape.

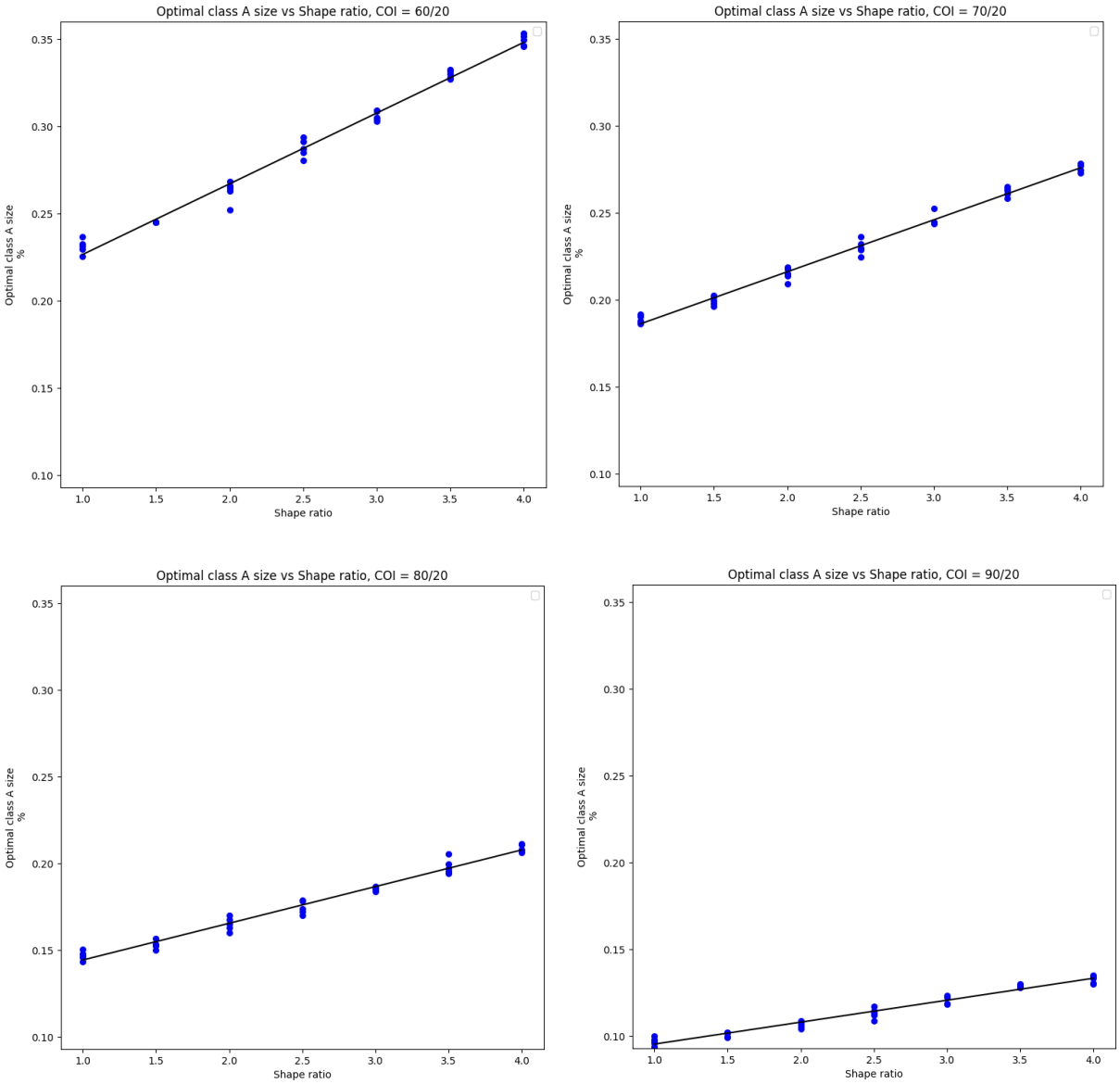



Figure 5.2: Optimal $l_A$ vs shape ratio with different ABC curves and triangular shape

In Figure 5.3 the regression lines are combined in the same plot, respectively considering a class A with a square shape and a class A with a triangular shape.

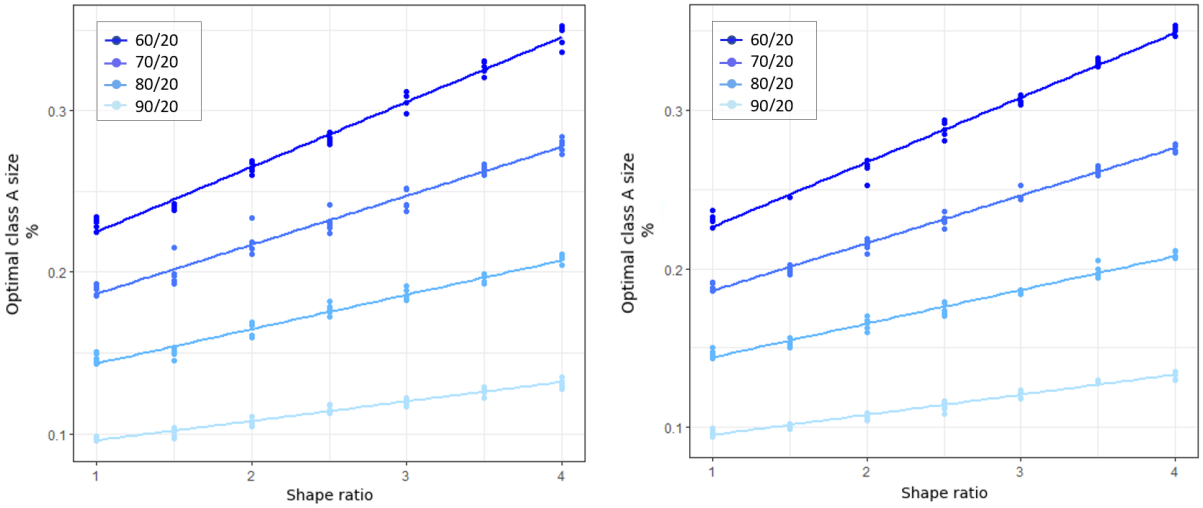



Figure 5.3: Optimal relationship between $l_A$ and shape ratio vs ABC curve with different class A shapes

A statistical analysis can be conducted to examine the effect of the various parameters on the optimal dimension of class A. Data from different ABC curves, grid sizes, and class A shapes have been combined for this analysis.
Table 5.5 presents the statistical significance of the parameters. It is observed that the grid size, along with the shape of class A, does not have a significant effect on determining the optimal value of $l_A$ (indicated by large p-values). However, the shape ratio does have a statistically significant positive effect. This means that as the shape ratio increases, the optimal value of $l_A$ slightly increases as well. More specifically, a unitary increase in the shape ratio leads to a 2.6 percentage point increase in the optimal value of $l_A$.
Additionally, the effect of the ABC curve is found to be statistically significant. Moving from an ABC curve with a ratio of 60/20 to an ABC curve with a ratio of 90/20 results in a reduction of the optimal value of $l_A$.

| Parameter | Coefficient | Standard deviation | t-statistic | p-value |
|---|---|---|---|---|
| Intercept | 0.2201 | 0.0032 | 69.8650 | 0.0000 * |
| Shape ratio | 0.0260 | 0.0007 | 38.5710 | 0.0000 * |
| Grid size | 0.0000 | 0.0000 | 0.455 | 0.6490 |
| Triangular shape | 0.0006 | 0.0014 | 0.423 | 0.673 |
| ABC 70/20 | -0.0547 | 0.0019 | -28.7130 | 0.0000 * |
| ABC 80/20 | -0.1107 | 0.0019 | -58.0490 | 0.0000 * |
| ABC 90/20 | -0.1723 | 0.0019 | -90.371 | 0.0000 * |

Table 5.5: Statistical significance of input parameters

Under the assumption that the shape ratio of the area dedicated to load storage can be chosen, it is important to determine the optimal value of the shape ratio that maximizes the system's performance in terms of average cycle time and density.
Considering grids of different sizes, the results in terms of average cycle time are presented in Figure 5.4. Each plot corresponds to a specific scenario, which is determined by the combination of a particular ABC curve and either a square or triangular shape for class A. It is observed that a square grid (shape ratio equal to 1) consistently yields the best performance across all scenarios.

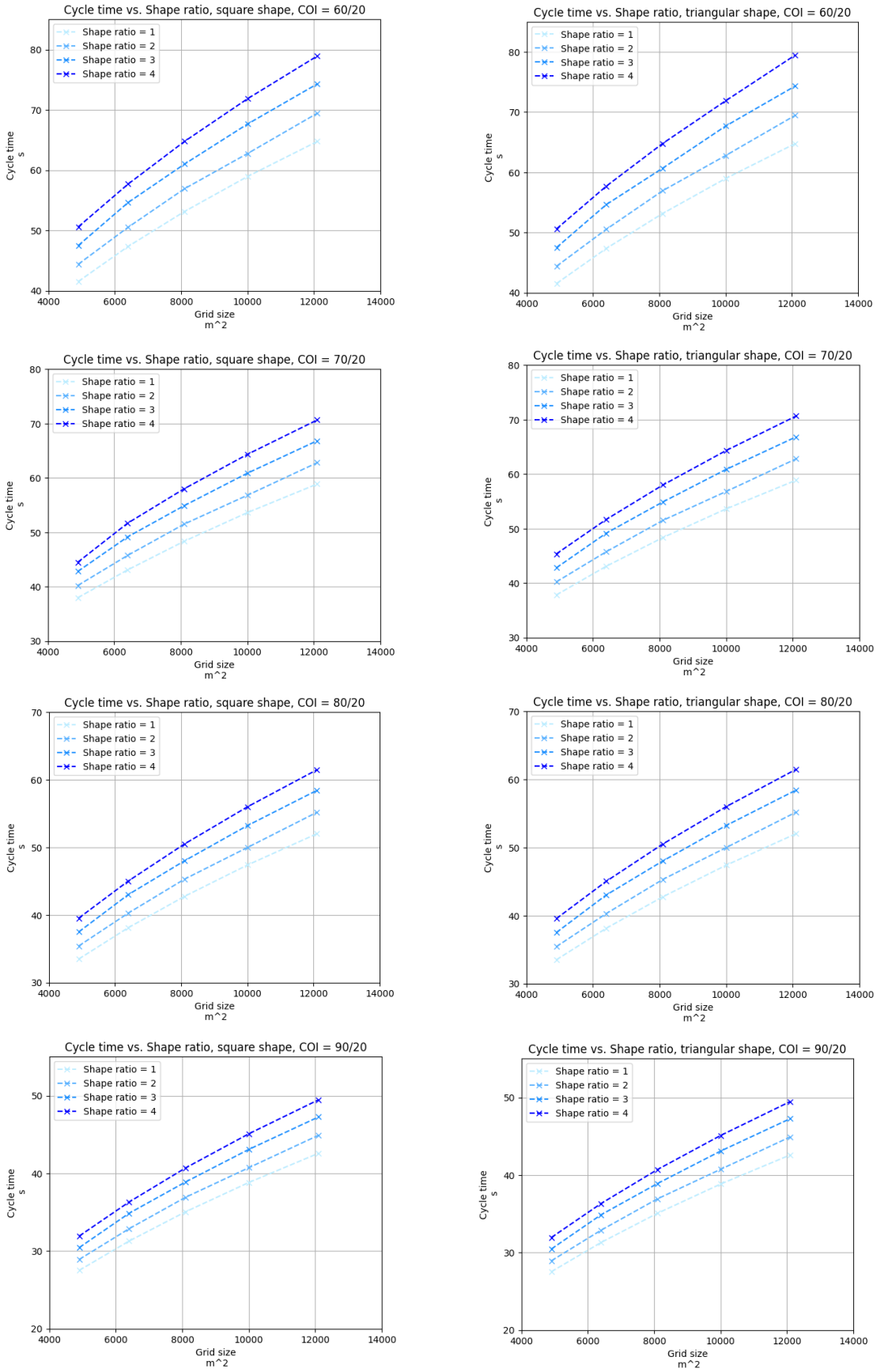



Figure 5.4: Average cycle time for different shape ratios

When considering the density of the grid, the class-based storage policy does not have an impact on the system's performance. The only parameter to be taken into consideration is the shape ratio of the grid. Figure 5.5 illustrates that the layout maximizing the density is characterized by a shape ratio of 2.
It is important to note that the curve for a shape ratio of 1 (a square grid) is not visualized because it yields the same density performance as a shape ratio of 4.
This finding suggests a trade-off between density and throughput. A shape ratio of 2 provides the highest density, indicating a more efficient utilization of the available storage space. However, it may come at the expense of slightly lower throughput.

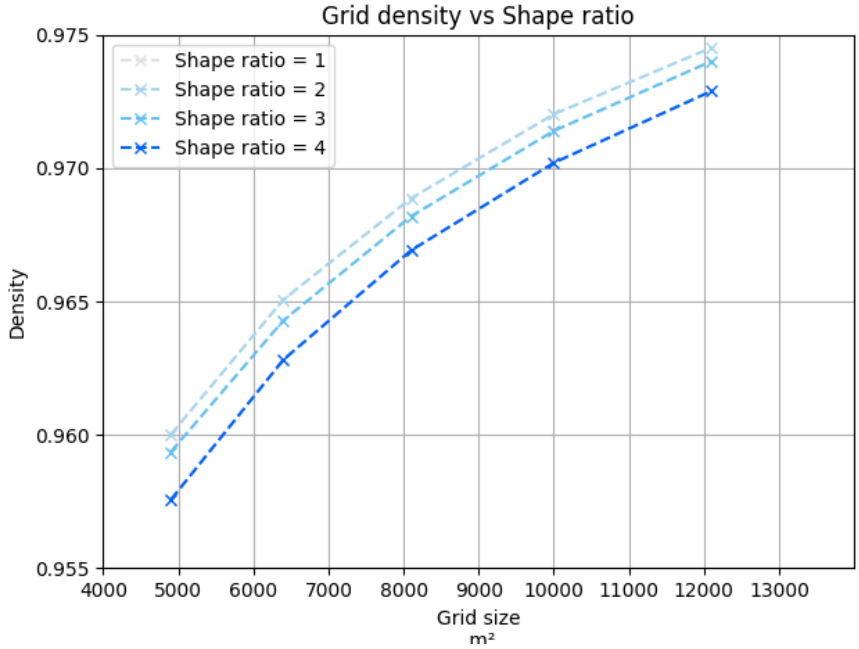



Figure 5.5: Density for different shape ratios

The shape of class A is the final input in the design procedure. The difference in terms of average cycle time between a square shape and a triangular shape is so small (less than 1 second) that it cannot be visualized effectively. However, the effect of the class A shape on the average cycle time has been analyzed by considering various grid dimensions for each possible system configuration. The findings, including the percentage of cases where the triangular shape results to be better and the average reduction in cycle time, are summarized in Table 5.6.

| Scenario | % Cases | Average reduction (s) |
|---|---|---|
| sr = 1, COI 60/20 | 64% | - 0.234 |
| sr = 1, COI 70/20 | 60% | - 0.242 |
| sr = 1, COI 80/20 | 63% | - 0.257 |
| sr = 1, COI 90/20 | 65% | - 0.251 |
| sr = 2, COI 60/20 | 73% | - 0.284 |
| sr = 2, COI 70/20 | 60% | - 0.233 |
| sr = 2, COI 80/20 | 54% | - 0.229 |
| sr = 2, COI 90/20 | 63% | - 0.286 |
| sr = 3, COI 60/20 | 80% | - 0.368 |
| sr = 3, COI 70/20 | 95% | - 0.525 |
| sr = 3, COI 80/20 | 75% | - 0.308 |
| sr = 3, COI 90/20 | 68% | - 0.314 |
| sr = 4, COI 60/20 | 2% | - 0.016 |
| sr = 4, COI 70/20 | 2% | - 0.102 |
| sr = 4, COI 80/20 | 94% | - 0.448 |
| sr = 4, COI 90/20 | 61% | - 0.279 |

Table 5.6: Triangular shape vs. Square shape performance

The lower average cycle time observed with a triangular shape can be attributed to its ability to accommodate a class A capacity that is closer to the required capacity. When defining the dimension of class A based on the optimal percentage of racks to be dedicated to this area, the result is rounded up to ensure that the actual capacity is at least equal to the required capacity.

When comparing the effects of increasing the class A dimension by one unit, it is evident that the increase in capacity is almost twice as significant for a square shape compared to a triangular shape. This discrepancy in capacity increase contributes to slightly higher cycle times and lower throughput in systems employing a class A with square shape.

It is important to note that designing a class A with a square shape is generally more straightforward and requires less effort compared to a triangular shape. This consideration becomes particularly relevant when making a choice between the two shapes, as it

presents a trade-off between design precision and design complexity. The square shape offers simplicity and ease of implementation, while the triangular shape provides a closer match to the required capacity, resulting in slightly improved performance in terms of average cycle time.

To summarize the main insights from the analysis for designing the system under a class-based storage policy:

- The implementation of the class-based storage policy does not affect the system's density performance. The density is only determined by the shape ratio.
- When allocating a storage area, choosing a shape ratio of 1 or 2 is generally recommended. A shape ratio of 1 prioritizes high throughput, while a shape ratio of 2 maximizes density.
- The optimal percentage of storage racks dedicated to class A items depends on the shape ratio and ABC curve, while the shape of class A itself does not significantly impact the optimal value.
- Designing a class A with a triangular shape can lead to slight improvements in average cycle time compared to a square shape. However, this improvement is not statistically significant, and other factors such as operational preferences, ease of implementation, and system compatibility should be considered when deciding on the class A shape.

# 6 | Random vs. Class-based

In order to assess the effectiveness of different storage policies, a comparison is conducted between the optimal system configurations with a class-based storage policy and the configuration with a random storage policy.

The case of the random storage policy consists in setting a percentage of racks dedicated for class A items ($l_A$) equal to 100%. Consequently, the associated probability of a requested load belonging to class A ($p_A$) is also 100%.
To analyze the effect of the storage policies, a triangular shape is adopted, leading to slight improvements in terms of average cycle time for the class-based storage policy.

This comparison is made considering all possible configurations in terms of shape ratio and ABC curve. By considering various grid sizes, the results are shown in Figure 6.1, where each subfigure takes into account a specific value of the shape ratio.
The graph clearly demonstrated that system configurations employing the class-based storage policy consistently outperformed the random storage configuration, regardless of the specific ABC curve characterizing the items or the shape ratio of the grid.

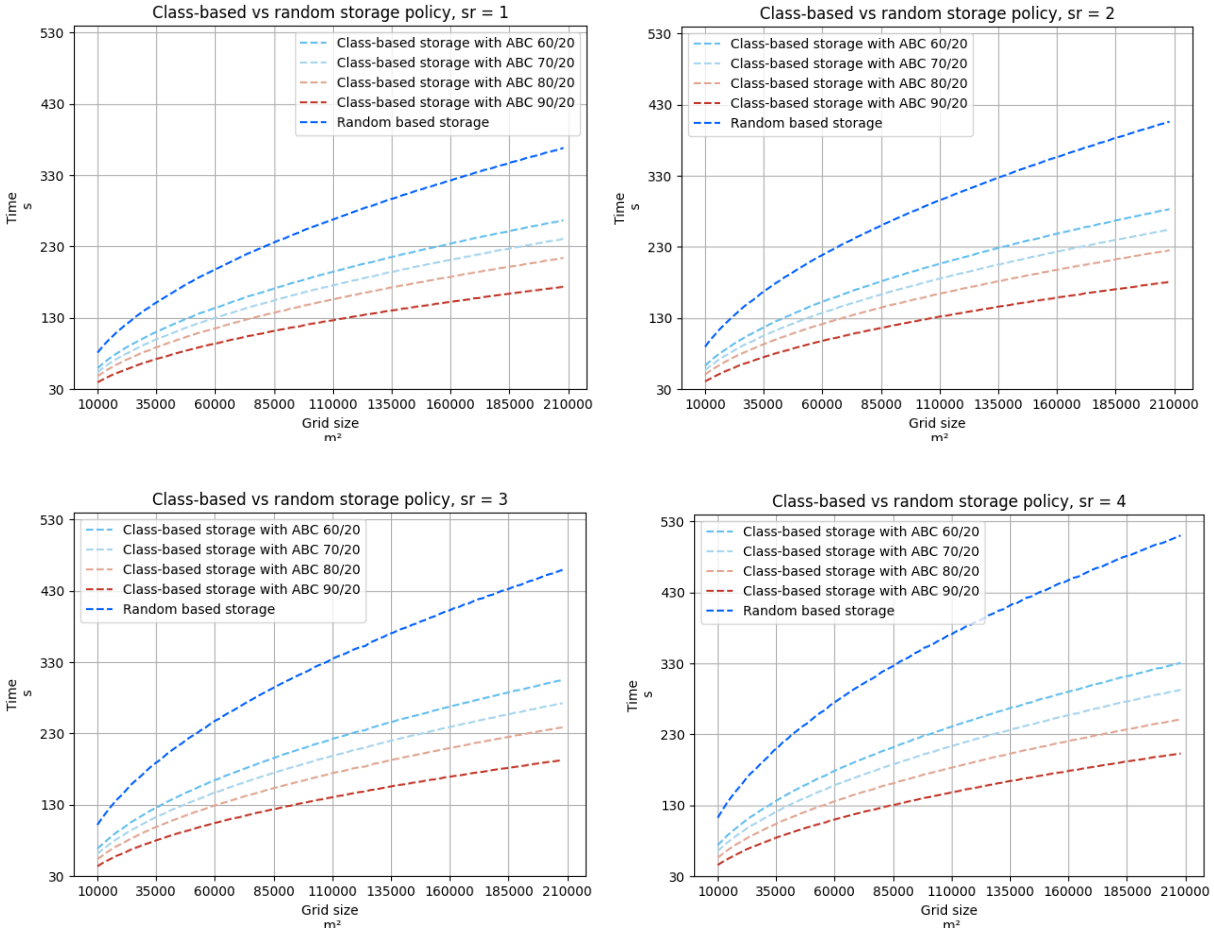



Figure 6.1: Class-based storage vs random storage policy

To provide more precise insights, Table 6.1 summarizes the reduction in average cycle time achieved by adopting the class-based storage policy compared to the random storage policy across different configurations.

| Scenario | Average reduction |
|---|---|
| sr = 1, ABC 60/20 | - 37.70% |
| sr = 1, ABC 70/20 | - 52.53% |
| sr = 1, ABC 80/20 | - 71.70% |
| sr = 1, ABC 90/20 | - 111.17% |
| sr = 2, ABC 60/20 | - 43.19% |
| sr = 2, ABC 70/20 | - 59.34% |
| sr = 2, ABC 80/20 | - 79.78% |
| sr = 2, ABC 90/20 | - 123.83% |
| sr = 3, ABC 60/20 | - 50.39% |
| sr = 3, ABC 70/20 | - 68.28% |
| sr = 3, ABC 80/20 | - 91.91% |
| sr = 3, ABC 90/20 | - 137.35% |
| sr = 4, ABC 60/20 | - 54.04% |
| sr = 4, ABC 70/20 | - 73.84% |
| sr = 4, ABC 80/20 | - 102.56% |
| sr = 4, ABC 90/20 | - 150.12% |

Table 6.1: Reduction average cycle time in class-based storage policy

Based on the findings, it is evident that the adoption of a class-based storage policy leads to greater benefits in terms of reducing the average cycle time as both the ABC curve and shape ratio increase.

It is important to acknowledge that, based on the assumptions made in this study, the choice of the storage policy does not have any impact on the overall system density. However, research conducted by Yugang Y. and De Koster (2010) has shown that in situations involving an infinite number of items, the required storage space in a class-based storage system is not equal to the average inventory levels. In fact, it is higher due to the fixed dedicated space for each class and the limited opportunity to share available space. Consequently, opting for a class-based storage policy results in lower density performance compared to a randomized allocation.

For future advancements, it is recommended to estimate the reduction and evaluate the trade-off between throughput and density.

# 7 | Comparison with literature

In this section, the improvements achievable with the implementation of the new configuration are presented with respect to the systems analyzed in the literature. These improvements are evaluated in terms of both density and average cycle time.

## Density

As already mentioned, the performance of the system in terms of density does not change whether a class-based storage policy or a random storage policy is implemented. This is true only under the underlying assumptions of this research.

The most comparable system in the mentioned literature is the virtual aisle configurations investigated by Yalcin et al. (2019b). They achieved densities between 0.89 and 0.94 with a capacity of the system equal on average to 1800 loads.
Considering the same capacity, the new configuration allows for densities respectively of 0.933, 0.936, 0.934 and 0.933 depending on the shape ratio of the grid. Therefore, the results are very similar.

The only configuration present in the literature outperforming the new proposed system in terms of density is the one investigated by Yalcin et al. (2019b) when the virtual aisle strategy is not adopted. The density achieved is equal to 0.99, but the system has very low performance in terms of throughput because simultaneous movements are not allowed.
Compared to RMF systems, our proposed configuration can store a similar number of loads at almost half the area. The multi-deep RMF system configuration by P. Yang et al. (2021) achieves a density of 0.5.

## Average cycle time

The virtual aisle configurations of Yalcin et al. (2019b) that they considered optimal achieve an average cycle time of 18.5 seconds. However, the configurations are not directly comparable as Yalcin et al. assume that loads can move with a speed of 2m/s, where we have assumed 1m/s, and their configuration has shape ratios equal to $sr = 6$ and $sr = 9$. Moreover, they considered 6 and 7 separate I/O locations distributed along the longest side.

Other configurations with a grid size of $2130m^2$ that they discard because of too poor performance have 3 and 4 I/O locations and a shape ratio equal to 1. These configurations result in average retrieval times respectively of 58 seconds and 63 seconds.
These results are still not directly comparable with the finding of this work because of the multiple I/O locations, allowing a reduction of the average travel distance.

With the aim to compare the results, it has been assumed that it is possible to model a square grid with multiple I/O locations simply by combining several smaller grids, referred to as instances, each one served by an I/O point.
When a number of I/O locations equal to 3 is taken into account the grid can be modeled combining 3 rectangular instances with $sr = 3$ and a length of the shortest dimension that is equal to 1/3 of the original one. When a number of I/O locations equal to 4 is considered, the grid can be modeled by combining 4 square instances with lengths of both dimensions equal to 1/2 of the original one. Considering a general grid with dimension $mxm$, Figure 7.1 shows the subdivision in the case of 3 I/O points, while the case with 4 I/O points is illustrated in Figure 7.2.

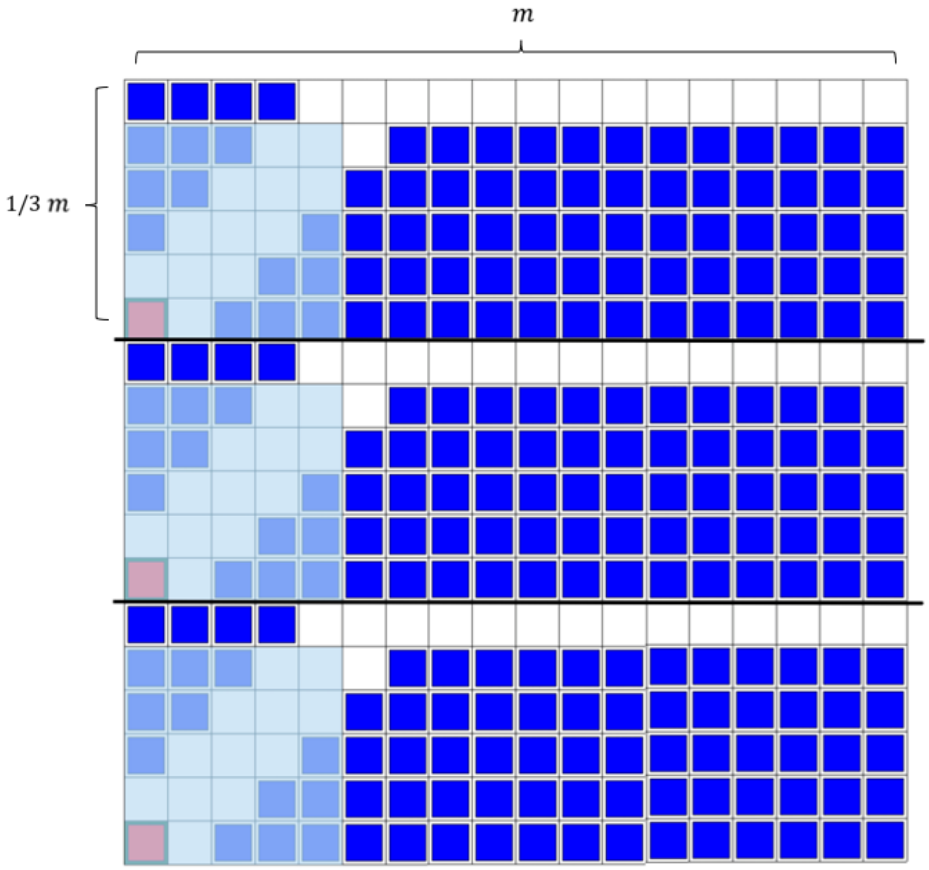



Figure 7.1: Subdivision of the grid with 3 I/O locations

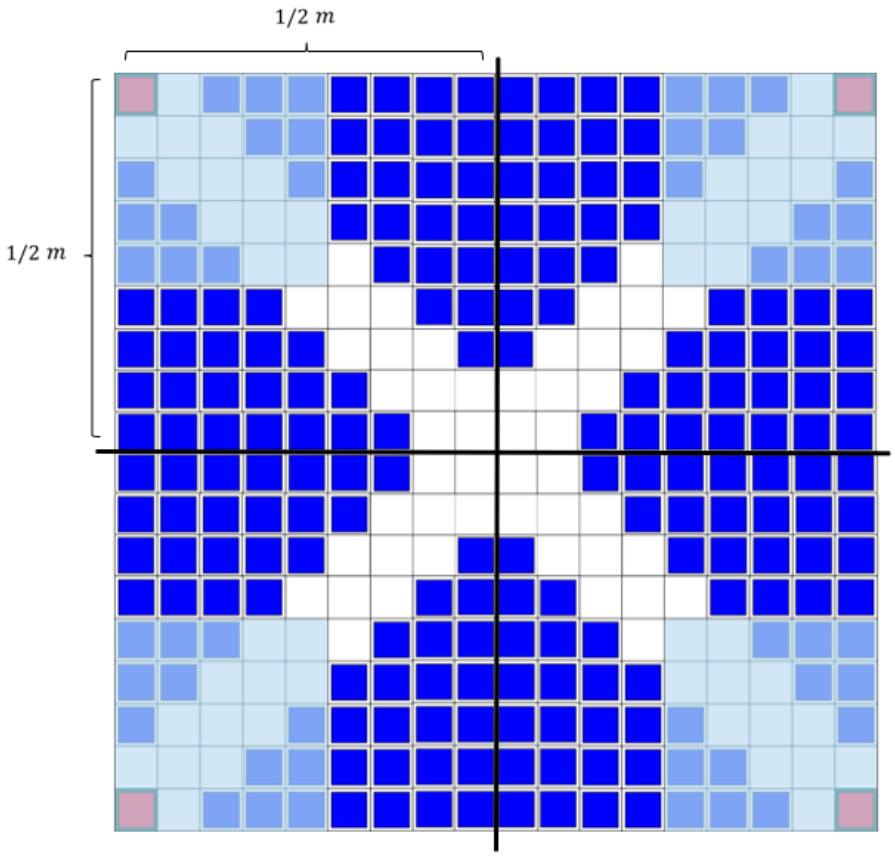



Figure 7.2: Subdivision of the grid with 4 I/O locations

Since loads are retrieved from all the sub-grids simultaneously, the performance of the overall grid in terms of average retrieval time coincides with the average cycle time of a smaller grid.

When the random storage policy is implemented the new configuration enables an average retrieval time of about 26.891 seconds and 19.004 seconds with respectively 3 and 4 I/O locations. The second result is very close to the one obtained by them with the configurations they considered optimal.
When the class-based is adopted as a storage policy in the new system configuration, it is possible to achieve average retrieval times far shorter, whatever ABC curve characterizes the stored items. The results are shown in Table 7.1.

| I/O points | ABC curve | Average cycle time (s) |
|---|---|---|
| 3 | 60/20 | 18.552 |
| 3 | 70/20 | 16.636 |
| 3 | 80/20 | 15.201 |
| 3 | 90/20 | 12.776 |
| 4 | 60/20 | 14.286 |
| 4 | 70/20 | 12.998 |
| 4 | 80/20 | 11.925 |
| 4 | 90/20 | 10.179 |

Table 7.1: Average cycle time with 3 and 4 I/O locations

# 8 | Conclusions and future developments

In this chapter, Section 8.1 provides a summary of the primary findings obtained. Section 8.2 addresses the limitations associated with the assumptions made for the purpose of this analysis, while Section 8.3 outline the potential future developments.

## 8.1. Conclusions

This work focuses on a new system configuration that has the potential to revolutionize e-commerce operations. The system leverages a new technology for material handling, the autonomous wheels recently developed by the company Wheel.me. By combining elements from Puzzle-based systems (PBS) and Robotic mobile fulfillment (RMF) systems, along with diagonal movements, this configuration shows significant improvements in terms of retrieval time, without significant reduction in density.

The previous research has demonstrated promising results. The devised movement policy and virtual aisle strategy ensure uninterrupted retrieval of loads within the system. Moreover, the utilization of the fishbone route significantly enhances system performance compared to traditional aisle configurations.
To further optimize performance, a class-based storage policy has been implemented. This approach does not impact the high-density achievable while simultaneously reducing cycle times. The most significant improvements in average cycle times deriving from the adoption of this storage policy are achieved when the SKUs are characterized by an ABC curve of 90/20.
Regardless of the ABC curve, the system's best performance in terms of throughput is obtained by designing a square storage area and dedicating a percentage of storage racks to class A items ranging between 10% and 23%.

Overall, the new system demonstrates significant improvements. However, it is essen-

tial to acknowledge certain limitations, which emphasize the need for further research to fully explore its potential.

## 8.2. Limitations

The proposed approach in this work has certain limitations and should be viewed as a starting point for future research, rather than a ready-to-implement solution.

- The first limitation is the position of the I/O location, which is considered fixed for the purpose of this analysis. Moreover, the presence of a single I/O point restricts the system from retrieving more loads at the same time. As already mentioned, to overcome this limitation, additional I/O locations could be introduced and the system could be modeled as a combination of multiple instances. The definition of the optimal combination is to be addressed.
- The proposed movement policy should be seen as a feature of the grid configuration rather than an optimized algorithm. It does not aim to minimize cycle time, travel distance, or load movement. Instead, it demonstrates the potential performance gains of less constrained movement.
- Another limitation comes from the assumption that the required storage space is equal to the average inventory level. In reality, when dealing with a finite number of items, the necessary storage space in a class-based storage system would be higher due to the fixed dedicated space for each class and the lower possibility to share the available space (Yugang Y. and De Koster (2010)).
- Finally, it is worth noting that no physical dynamic moveable rack system with autonomous wheels has been implemented yet, as autonomous wheels are a relatively new technology, while RMF systems and some PBS systems with conveyor belts have been used in e-commerce warehousing. This is why further research is needed to address operational aspects such as designing algorithms for communication between a large number of self-driving racks with autonomous wheels.

## 8.3. Possible future developments

This study examines a system configuration characterized by a single I/O point, which leads to high average travel distances within large grids, resulting in reduced throughput. To mitigate this issue, the entire grid can be conceptualized as a collection of subgrids,

each served by its own dedicated I/O location. This has been already shown in Chapter 7.
However, it is necessary to determine an optimal subdivision of the original grid for each configuration, taking into account the desired number of additional I/O points.
By identifying the most suitable combination, the performance of the system, which can be evaluated by summing those of the single instances, can be significantly enhanced.

Other space for future exploration lies in the concept of a dynamic I/O location. Instead of a fixed I/O point, the position of the I/O location can be adjusted dynamically based on the load being retrieved.
This dynamic positioning enables the opening of straight virtual aisles, depending on the specific location of the I/O point at any given time. By managing the I/O location based on the load requirements, it becomes possible to optimize retrieval operations and minimize travel distances within the grid.

In the longer term, the research could investigate how diagonal aisles can be opened at any angle with respect to the I/O location, not only 45°, as modeled in this work.
In this sense, it is required to review the constraints of the movement of the loads in the grid and the allocation of escorts.

In addition to system configuration optimization, further research efforts could be directed toward refining the operational management of the system. This entails developing strategies for determining the optimal sequence in which loads are retrieved.
By finding the optimal order so that the next load to be picked up is not moved far but is located in close proximity to the I/O point, the overall retrieval process can be optimized, resulting in improved efficiency and reduced travel distances.

# 9 | Bibliography

Bhaskar, S. et al. (2020). 'At the epicenter of COVID-19: the tragic failure of the global supply chain for medical supplies'. *Frontiers in public health 8*

Bortolini, M. et al. (2015). 'Diagonal cross-aisles in unit load warehouses to increase handling performance'. *International Journal of Production Economics 170*

Bortolini et al. (2018) 'Class-Based storage warehouse design with diagonal cross-aisle'. *Scientific Journal of Logistics, 14.1*

Boysen, N., D. Briskorn and S. Emde (2017). 'Parts-to-picker based order processing in a rack-moving mobile robots environment'. *European Journal of Operational Research 262.2*

Boysen, N., R. De Koster and F. Weidinger (2019). 'Warehousing in the e-commerce era: A survey'. *European Journal of Operational Research 277.2*

Bukchin, Y. and T. Raviv (2020). 'Optimal retrieval in puzzle-based storage systems with simultaneous load and block movement' *Transportation Science 57.2*

De Koster et al. (2007). 'Design and control of warehouse order picking: A literature review' *European Journal of Operational Research, 182.2*

Garnett, P., B. Doherty and T. Heron (2020). 'Vulnerability of the United Kingdom's food supply chains exposed by COVID-19'. *Nature Food 1.6*

Gharehgozli, A. and N. Zaerpour (2020). 'Robot scheduling for pod retrieval in a robotic mobile fulfillment system'. *Transportation Research Part E: Logistics and Transportation Review 142.*

Graves et al. (1977). 'Note—Deriving the Optimal Boundaries for Class-Based Automatic Storage/Retrieval Systems'. *Management Science, 35.12*

Gu et al. (2007). 'Research on warehouse operation: A comprehensive review'. *European Journal of Operational Research, 177.1*

Gue, K.R. (2006). 'Very high density storage systems'. *IIE Transactions 38.1*

Gue, K.R. and B.S. Kim (2007). 'Puzzle-based storage systems'. *Naval Research Logistics 54.5*

Gue, K.R. and R.D. Meller (2009). 'Aisle configurations for unit-load warehouses'. *IIE transactions 41.3*

Gue, K.R., K. Furmans et al. (2014). 'GridStore: A puzzle-based storage system with decentralized control'. *IEEE Transactions on Automation Science and Engineering 11.2*

Guthrie, C., S. Fosso-Wamba and J.B. Arnaud (2021). 'Online consumer resilience during a pandemic: An exploratory study of e-commerce behavior before, during and after a COVID-19 lockdown'. *Journal of Retailing and Consumer Services 61*

Hausman et al. (1976). 'Optimal Storage Assignment in Automatic Warehousing Systems'. *Management Science, 22.6*

Keung, K.L. et al. (2020). 'Cloud-Based Cyber-Physical Robotic Mobile Fulfillment Systems: A Case Study of Collision Avoidance'. *IEEE Access 8*

Kim, H., C. Pais and Z.-J.M. Shen (2020). 'Item assignment problem in a robotic mobile fulfillment system'. *IEEE Transactions on Automation Science and Engineering 17.4*

Kota, V.R., D. Taylor and K.R. Gue (2015). 'Retrieval time performance in puzzle-based storage systems'. *Journal of Manufacturing Technology Management 26.4*

Lamballais, T., D. Roy and M.B.M. De Koster (2017). 'Estimating performance in a Robotic Mobile Fulfillment System'. *European Journal of Operational Research 256.3*

Merschformann, M. et al. (2019). 'Decision rules for robotic mobile fulfillment systems'.

*Operations Research Perspectives 6*

Mirzaei, M., R.B.M. De Koster and N. Zaerpour (2017). 'Modelling load retrievals in puzzle-based storage systems'. *International Journal of Production Research 55.21*

Rohit, Kota V, G Don Taylor and K.R. Gue (2010). 'Retrieval time performance in puzzle-based storage systems'. *IIE Annual Conference. Proceedings*

Rosenblatt and Eynan (1989). 'Note—Deriving the Optimal Boundaries for Class-Based Automatic Storage/Retrieval Systems'. *Management Science, 35.12*

T., Lamballais, D. Roy and R. De Koster (2020). 'Inventory allocation in robotic mobile fulfillment systems'. *IISE Transactions 52.1*

Thonemann and Brandeau (1998) 'Optimal Storage Assignment Policies for Automated Storage and Retrieval Systems with Stochastic Demands'. *Management Science, 44.1*

Wurman, P, R. D'Andrea and M. Mountz (2008). 'Coordinating hundreds of cooperative, autonomous vehicles in warehouses'. *AI magazine 29.1*

Xie, L. et al. (2021). 'Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems'. *European Journal of Operational Research 288.1*

Yalcin, A., A. Koberstein and K.-O. Schocke (2019a). 'An optimal and a heuristic algorithm for the single-item retrieval problem in puzzle-based storage systems with multiple escorts'. *International Journal of Production Research 57.1*

Yalcin, A., A. Koberstein and K.-O. Schocke (2019b). 'Motion and layout planning in a grid-based early baggage storage system: Heuristic algorithms and a simulation study'. *OR Spectrum 41.3*

Yang, P., G. Jin and G. Duan (2021). 'Modelling and analysis for multi-deep compact robotic mobile fulfilment system'. *International Journal of Production Research*

Yang, Xiuqing et al. (2021). 'Non-traditional layout design for robotic mobile fulfillment system with multiple workstations'. *Algorithms 14.7*

Yu, Y. et al. (2017). 'Optimal algorithm for minimizing retrieval time in puzzle-based storage system with multiple simultaneously movable empty cells.' *Tech. rep. Working Paper Erasmus University, Rotterdam.*

Yugang Y. and De Koster (2010). 'Class-based Storage With a Finite Number of Items'. *IMHRC Proceedings 34.11*

Yunfeng, M, C. Haoxun and Y. Yugang (2021). 'An efficient heuristic for minimizing the number of moves for the retrieval of a single item in a puzzle-based storage system with multiple escorts'. *European Journal of Operational Research*

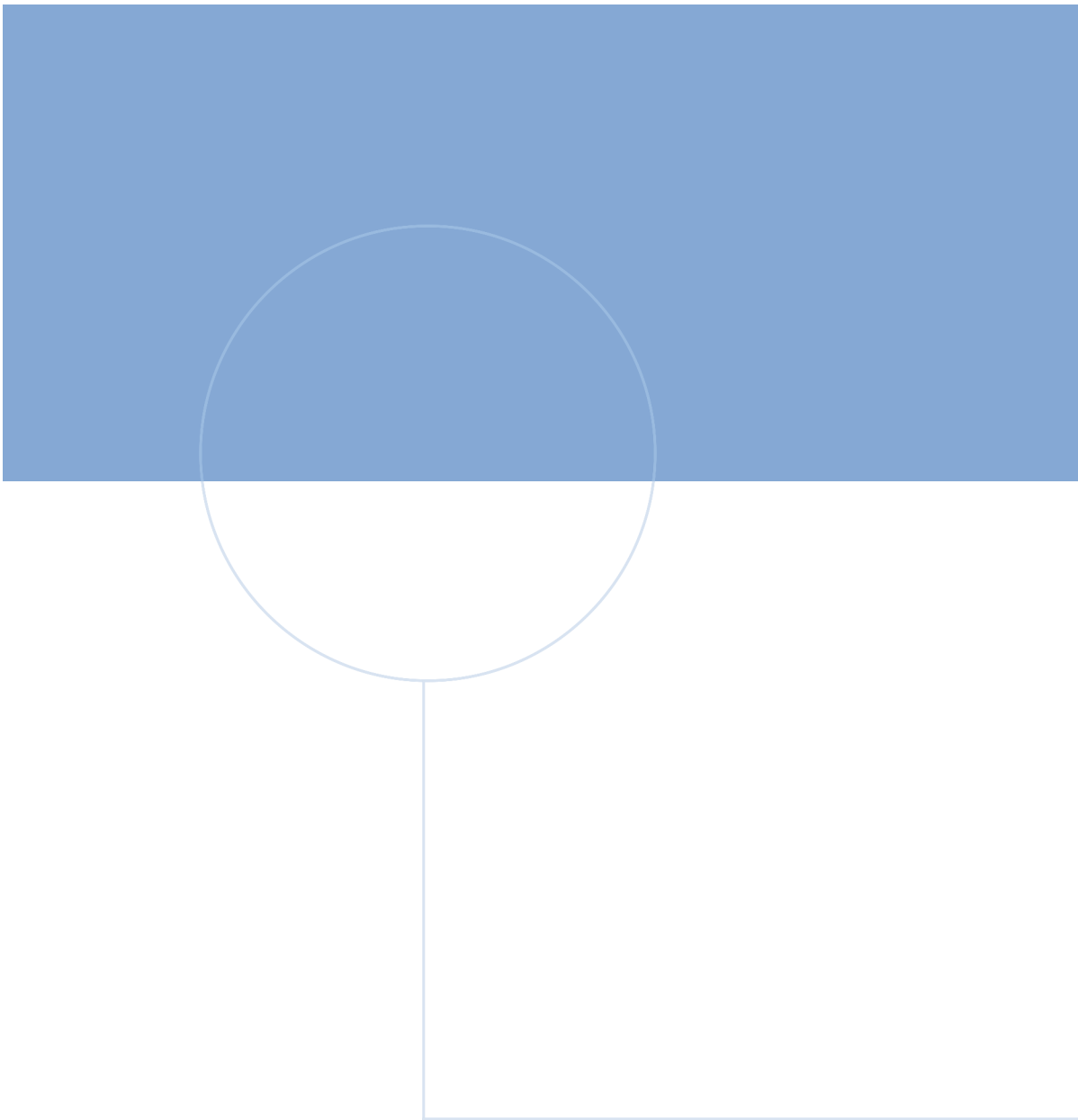
Zaerpour, N., Y. Yu and R. De Koster (2017a). 'Response time analysis of a live-cube compact storage system with two storage classes'. *IISE Transactions 49.5*

Zaerpour, N., Y. Yu and R. De Koster (2017b). 'Small is beautiful: A framework for evaluating and optimizing live-cube compact storage systems'. *Transportation Science 51.1*

Zou, B. et al. (2017). 'Assignment rules in robotic mobile fulfilment systems for online retailers'. *International Journal of Production Research 55.20*

# List of Figures

# List of Tables



NTNU
Norwegian University of
Science and Technology

POLITECNICO
MILANO 1863