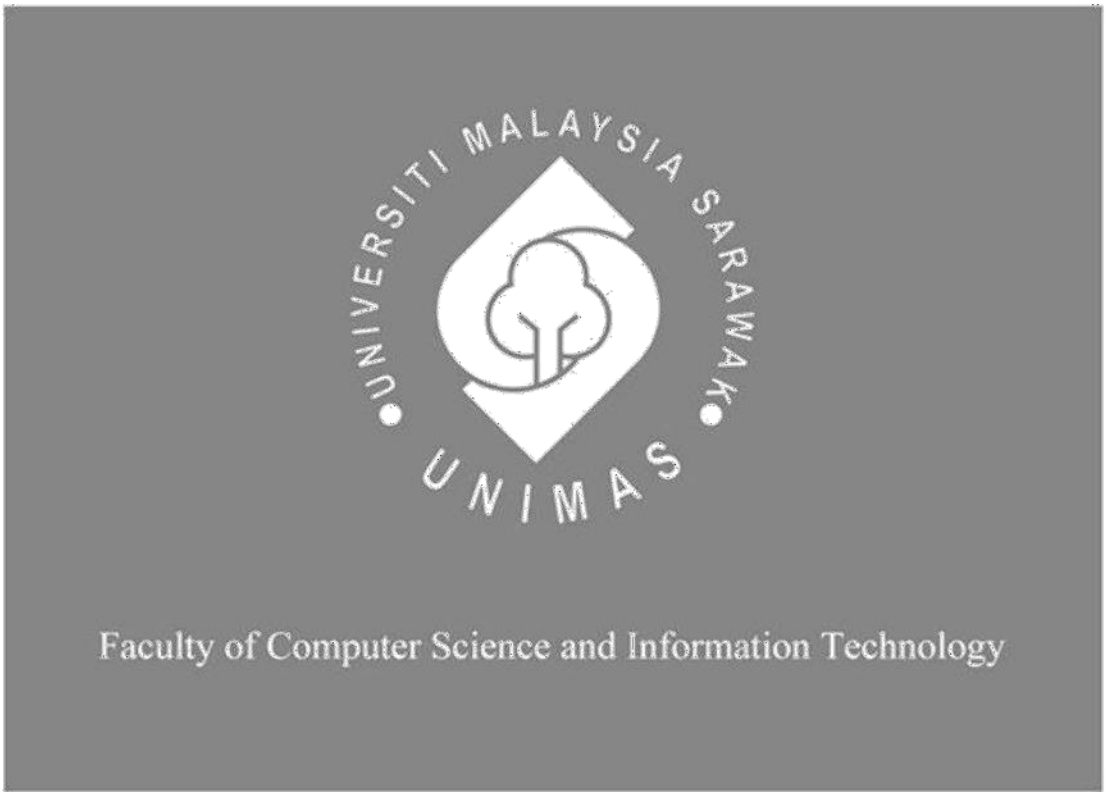


OPTIMIZATION OF CROWD EVACUATION SIMULATION IN TEACHING AND LEARNING FACILITIES

HAMIZATULWAFAA BINTI ABDUL GHANI

Bachelor of Software Engineering with Honours

2025

OPTIMIZATION OF CROWD EVACUATION SIMULATION IN TEACHING AND LEARNING FACILITIES

HAMIZATULWAFAA BINTI ABDUL GHANI

This project is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science with Honours (Software Engineering)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2025

PENGOPTIMUMAN SIMULASI PEMINDAHAN ORANG RAMAI DI KEMUDAHAN PENGAJARAN DAN PEMBELAJARAN

HAMIZATULWAFAA BINTI ABDUL GHANI

Projek ini merupakan salah satu keperluan untuk Ijazah Sarjana Muda Sains Komputer dengan Kepujian
(Kejuruteraan Perisian)

Fakulti Sains Komputer dan Teknologi Maklumat
UNIVERSITI MALAYSIA SARAWAK
2025

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE **OPTIMIZATION OF CROWD EVACUATION SIMULATION IN TEACHING AND LEARNING FACILITIES**

ACADEMIC SESSION: 2024/2025

HAMIZATULWAFAA BINTI ABDUL GHANI

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
√	UNRESTRICTED	

Validated by

(AUTHOR'S SIGNATURE)

(SUPERVISOR'S SIGNATURE)

Permanent Address

30, LORONG PERMATANG BADAK
BARU 7, TAMAN BUKIT RANGIN,
25150, KUANTAN, PAHANG

Date: 21st JUNE 2025

Date: _____

DECLARATION

I hereby declare that the work in this thesis entitled “Optimization of Crowd Evacuation Simulation in Teaching and Learning Facilities” is my own original work except as cited in the references. It has not been submitted previously or concurrently for any other degree or diploma at this or any other institution.

I confirm that all sources of information used in this thesis have been duly acknowledged.

This thesis is submitted in partial fulfilment of the requirements for the degree of Bachelor of Software Engineering with Honours at Universiti Malaysia Sarawak.

HAMIZATULWAFAA BINTI ABDUL GHANI 22 JUNE 2025

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to Madam Hamizan binti Sharbini, my final year project supervisor, for her unwavering support, valuable guidance, and continuous encouragement throughout the completion of this project. Her dedication and insightful advice have been instrumental in shaping this work.

I would also like to extend my heartfelt appreciation to my beloved parents for their unconditional love, endless support, and constant belief in my abilities. Their presence and encouragement have been a source of strength for me throughout this journey.

A sincere thank you goes to myself for preserving through challenges, for knowing when to pause, and for continuing to move forward despite the obstacles faced.

I am also grateful to my friends for their support, assistance, and motivation along the way. A special note of appreciation to Amizul Faizzien bin Shahrul Azhar, my partner, for his continuous belief in me, especially during times when I struggled to believe in myself. His unwavering encouragement and emotional support played a significant role throughout this journey.

Last but not least, I would like to thank everyone who has, directly or indirectly, contributed to the successful completion of this project. Your kindness, encouragement, and support have meant a great deal to me.

ABSTRACT

This study explores the optimization of crowd evacuation in teaching and learning facilities at Universiti Malaysia Sarawak (UNIMAS) using Particle Swarm Optimization (PSO), investigating whether PSO can enhance evacuation efficiency by reducing evacuation times and managing congestion, particularly in high-density scenarios or when exits are blocked. A crowd evacuation simulation system was developed using MATLAB, incorporating PSO to dynamically optimize evacuation routes based on building layouts and agent densities. The study tested three exit configurations: three exits open, two exits with one blocked, and one exit with two blocked, and also included slow-moving agents to simulate individuals with reduced mobility, such as the elderly. Results show that PSO consistently reduced evacuation times compared to non-optimized scenarios, with the most significant improvements observed in the one-exit scenario for 130 agents with 10 slow-moving agents, where evacuation times were reduced from 58.28 seconds in non-PSO simulations to 26.25 seconds with PSO. In scenarios with two exits, PSO reduced evacuation time from 23.75 seconds to 12.44 seconds for 130 agents, and with three exits, it decreased evacuation times from 14.26 seconds to 6.17 seconds for 10 agents. PSO also mitigated delays caused by slow agents, improving evacuation flow even in constrained environments, thus highlighting its effectiveness in optimizing crowd evacuation strategies in educational settings and offering potential applications for campus safety and emergency preparedness.

ABSTRAK

Kajian ini meneroka pengoptimuman pemindahan orang ramai di kemudahan pengajaran dan pembelajaran di Universiti Malaysia Sarawak (UNIMAS) menggunakan Pengoptimuman Swarm Partikel (PSO), dengan menyiasat sama ada aplikasi PSO dapat meningkatkan kecekapan pemindahan dengan mengurangkan masa pemindahan dan menguruskan kesesakan, terutamanya dalam senario kepadatan tinggi atau apabila laluan keluar disekat. Sistem simulasi pemindahan orang ramai telah dibangunkan menggunakan MATLAB, yang menggabungkan PSO untuk mengoptimumkan laluan pemindahan secara dinamik berdasarkan susun atur bangunan dan kepadatan agen. Kajian ini menguji tiga konfigurasi laluan keluar: tiga laluan keluar dibuka, dua laluan keluar dengan satu disekat, dan satu laluan keluar dengan dua disekat, serta memasukkan agen yang bergerak perlahan untuk menggambarkan individu dengan mobiliti terhad, seperti warga emas. Keputusan menunjukkan bahawa PSO secara konsisten mengurangkan masa pemindahan berbanding senario tanpa pengoptimuman, dengan peningkatan yang paling ketara diperhatikan dalam senario satu laluan keluar untuk 130 agen dengan 10 agen perlahan, di mana masa pemindahan dikurangkan dari 58.28 saat dalam simulasi tanpa PSO kepada 26.25 saat dengan PSO. Dalam senario dengan dua laluan keluar, PSO mengurangkan masa pemindahan dari 23.75 saat kepada 12.44 saat untuk 130 agen, dan dengan tiga laluan keluar, ia mengurangkan masa pemindahan dari 14.26 saat kepada 6.17 saat untuk 10 agen. PSO juga mengurangkan kelewatan yang disebabkan oleh agen bergerak perlahan, memperbaiki aliran pemindahan walaupun dalam persekitaran yang terhad, dengan menonjolkan keberkesanannya dalam mengoptimumkan strategi pemindahan orang ramai di persekitaran pendidikan dan menawarkan potensi aplikasi untuk keselamatan kampus dan kesiapsiagaan kecemasan.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 : INTRODUCTION

1.1 Introduction

During 1995 Kennedy and Eberhart proposed one computational technique named Particle Swarm Optimization (PSO) algorithm. This technique was inspired by the movements of a flock of birds flying or a school of fish that moves together in a group to achieve the same goal. For example, when one of the birds is randomly flying and searching for food, all birds that fly together can also get the benefit from the discovery of food. In PSO, the movement of a flock of birds can be interpret into a group of occupants or students in UNIMAS navigates through a fire outbreak's solution realm to find the best possible exit, especially in the lecture room settings. By applying this technique, each group will be able to adjust their own position based on their best solution in solving complex optimization problems such as determining the most efficient evacuation plan in lecture room in UNIMAS.

According to Srinvias et al. (2023) one of the MATLAB primary capabilities is modelling and simulation. This includes optimizing system performance, visualizing data and generate code from models. In developing an optimize crowd evacuation simulation in teaching and learning facilities at UNIMAS, MATLAB programming is the most efficient tools to use as it has the ability to develop advanced algorithms based on the complex data sets, built in functions for optimization, and visualization tools. Therefore, with its strong computational capabilities and extensive optimization toolboxes, MATLAB serves as the ideal tools for implementing PSO for this project.

At Universiti Malaysia Sarawak (UNIMAS), with its diverse campus infrastructure and dynamic academic community, optimizing crowd evacuation in teaching and learning facilities presents a multifaceted challenge that requires innovative solutions from many aspects. The urgent need to optimize crowd evacuation arises from the imperative to ensure the swift and

safe evacuation of students and staffs from affected buildings during emergencies. Having an optimized evacuation strategy can result in minimizing evacuation times and maximizing safety of individuals. This also can significantly reduce the risk of injuries or fatalities in emergency situations, thereby safeguarding the UNIMAS community and preserving lives.

In addition, recent statistics from the Sarawak Fire and Rescue Department stated that there have been several emergency calls in April 2024 alone, with fires affecting buildings being among them. Due to the significant increase, this study will explore and implement various optimization algorithms and delve into the intricate interplay between crowd evacuation simulation, optimization algorithms and data analysis. By harnessing MATLAB programming and the particle swarm algorithm as the primary computational tool, this research will be able to provide actionable insights and recommendations for enhancing campus safety and emergency preparedness at Universiti Malaysia Sarawak.

1.2 Problem Statement

It is important to guarantee the safe evacuation of individuals including students and occupants during emergency cases at UNIMAS, specifically in lecture room settings. However, traditional evacuation procedures often overlook individual differences in demographic factors such as height, gender, and age, which can potentially lead to suboptimal outcomes. To address this obstacle, this project aims to develop and implement a crowd optimization system using MATLAB programming and the particle swarm algorithm technique. By considering demographic factors in evacuation modelling, we seek to enhance the efficiency and safety of evacuation procedures in lecture rooms. This project will not only contribute to the advancement of crowd optimization techniques but also provide valuable insights for improving emergency preparedness and response in educational environments at UNIMAS.

1.3 Project Scope

The scope of this project is focussing on optimizing the crowd evacuation simulation system in teaching and learning facilities with multiple exits at UNIMAS. By using MATLAB, which serves as an ideal tool for implementing PSO, the emergency strategic plan will be enhanced, making it easier for everyone to review and understand the nearest emergency route during an evacuation. The scope of the project is as follows:

a) This crowd optimization system is developed for teaching and learning facilities at UNIMAS.
b) This crowd optimization system only utilizes MATLAB's data analysis capabilities to analyze and interpret relevant data sets.
c) This crowd optimization system is only accessible to occupants to view the data and accessible to students to view the evacuation plan.

1.4 Aims and Objectives

Crowd optimization simulation system in teaching and learning facilities at UNIMAS aims to help the occupants and students to move out of the building with promptness and caution through the understanding and application of optimization algorithms and data analysis using MATLAB programming. Specifically, the objectives of this project are:

1. To design simulation evacuation layout for educational settings.
2. To apply particle swarm optimization (PSO) in crowd evacuation simulation based on the crowd demographics factors and obstacles constraints.
3. To analyze evacuation time based on non-optimization and optimization evacuation time in lecture rooms.

1.5 Brief Methodology

The software methodology used for the development of a crowd optimization evacuation system in teaching and learning facilities—specifically for lecture rooms in UNIMAS—is the Agile methodology. Agile is an iterative and incremental development model that emphasizes flexibility, continuous feedback, and close collaboration between developers and stakeholders. Unlike traditional linear models, Agile divides the development process into smaller cycles known as *sprints*, where each sprint involves planning, design, implementation, testing, and review. This approach allows for frequent reassessment and adaptation of requirements, ensuring the system evolves based on user feedback and changing needs. By continuously refining the simulation through multiple iterations, Agile supports the development of a more accurate, reliable, and responsive evacuation system tailored to the dynamic conditions of real emergency scenarios (Stoica et al., 2013).

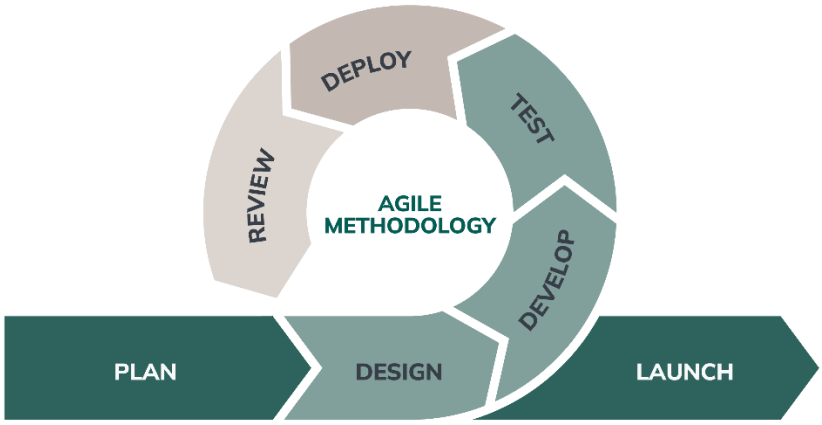


Figure 1.1: Agile methodology phase

1.6 Project Schedule

Creating a project planning with key milestones ensures that the development process remains organized, and progress can be tracked effectively. Table 1.1 below is the project planning timeline to ensure the entire development process will remain organized:

Table 1.1: Gantt Chart for FYP 1 and FYP 2 Project Schedule

	Final Year Project 1 (2024)					Final Year Project 2 (2025)				
Tasks	Mar	Apr	May	June	July	Mar	Apr	May	June	July
Preparation phase										
Full proposal submission										
Chapter 1: Introduction										
Chapter 2: Literature Review										
Chapter 3: Requirement Analysis & Design										
Submission of FYP 1 final report										
Chapter 4: Implementation										
Chapter 5: Testing										
Chapter 6: Conclusion and Further Work										
Submission of FYP 2 final report										

1.7 Significance of the Project

To summarize, this project holds significant importance due to its potential to significantly improve safety measures and optimize resource allocation during emergency situations on campus. Through the application of advanced computational techniques, optimization methodologies, and rigorous data analysis, we aspire to devise tailored solutions for crowd evacuation within the teaching and learning facilities at Universiti Malaysia

Sarawak. Ultimately, our aim is to cultivate a campus environment that prioritizes safety, resilience, and well-being, thereby fostering an optimal setting for academic pursuit and community engagement.

1.8 Expected Outcome

By the end of this project, we expected the development of a crowd optimization system specifically designed for lecture room settings in UNIMAS evacuation scenarios successfully implemented using MATLAB programming and the particle swarm algorithm. This system is designed to minimize evacuation time and congestion by integrating demographic factors such as height, gender, and age. Additionally, this project will yield valuable insights into the influence of demographic factors on evacuation behaviour, contributing to a deeper understanding of human dynamics in emergency situations. Ultimately, the project outcomes will not only advance the field of crowd optimization but also offer practical recommendations for enhancing evacuation procedures in educational settings, thereby fostering safer learning environments for students and occupants.

1.9 Conclusion

In conclusion, the optimization of crowd evacuation simulation system will ensure the safety of every student, staffs, and occupant at UNIMAS, especially in the teaching and learning facilities during any emergency situations. The lecture room settings that feature multiple exits need a strategic evacuation plan to guarantee each individual able to exit the affected building in a shorter time. The project utilizes MATLAB to implement the Particle Swarm Optimization (PSO) algorithms in order to improve the current evacuation plans by providing a clearer and nearest evacuation route. The improved simulation system will consider demographic factors such as height, gender, and age, which can lead to crowd congestion and inefficient evacuation routes. By having an effective evacuation plan, this project goal will be

achieved simultaneously guaranteed the safety of all students and occupants in teaching and learning facilities in UNIMAS. The simulation of evacuating also should be concentrated on raising awareness and exploring the space constraint during evacuation.

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

This literature review is written to provide a detailed analysis of the application of Particle Swarm Optimization (PSO) as an optimization technique for crowd evacuation simulation scenarios. This optimization technique has been chosen due to its simplicity, efficiency and its ability to perform in diverse and complicated situations. By exploring the principles, advantages and disadvantages, and practical implementations of PSO, this literature review will thoroughly be explaining the reason PSO is an effective model for optimizing crowd evacuation strategies particularly in fire outbreak scenarios compared to other simulation models.

The chapter is written by beginning with an overview of various crowd optimization simulation models, covering their principles as well as comparative advantages and disadvantages. It was then followed by the detailed workings of PSO, including its principles and its specific applications in crowd evacuation simulation. Finally, this literature review will include a practical example to demonstrate the effectiveness of PSO's, followed by a summary of the whole chapter and recommendation for future research directions.

2.2 Overview of Crowd Simulations Models

Recently, computer-based modelling and simulation technologies have recently evolved to support in the study of crowd dynamics such as crowd behaviors under normal and emergent scenarios. This study proposed some crowd simulations models that are commonly used in research to study or mimic the dynamics of crowds. Despite several existing research efforts and applications in crowd modelling and simulation, it is still a young and emerging

area. Thus, there is a need for a comprehensive survey and comparison of crowd modelling and simulation models.

2.2.1 Agent-Based Model

Agent-based model (ABM) is a computational technique used to study complex social and crowd scenarios by simulating distinct individuals, known as agents. These agents have characteristics that allow them to perceive their environment and interact with others (Salgado & Gilbert, 2013). ABM facilitates individual behaviours and social interactions within a dynamic environment, closely resembling human behaviours in emergencies. However, ABM has disadvantages, such as the need for significant computational power and time, making it challenging to scale for large crowds or extensive environments. Additionally, the complexity of ABM complicates validation and result interpretation, making it essential for careful consideration when implementing ABM for crowd evacuation systems (Zia & Ferscha, 2020).

According to a case study of simulation by Kasereka et al. (2018) there are four parameters that allowed researchers to make an effective evaluation of their proposed model:

a) **The total of people alive (TV):** represents the number of people having a potency superior to 0 and who have left the building.

$$TV = \sum av \quad (1)$$

Where ***av*** refers to the alive evacuate agent

b) **Total deaths (TM):** represents the number of persons with a potency less than or equal 0.

$$TM = \sum ag - \sum av \quad (2)$$

Where ***ag*** is the agent in general and ***av*** refers to the alive evacuate agent.

c) **Average potency of the alive persons (MP):** represents the average potency of the evacuated alive agents (who were able to leave the building). It is calculated as follows:

$$MP = \frac{\sum_{i=1}^{n} pav_i}{\sum av} \quad (3)$$

Where ***pav*** is the potency of alive evacuee agent and ***av*** refers to the alive evacuee agent.

d) **Average time taken to exit (MT):** means the average time taken by an evacuated agent to leave the building. It is calculated as follows:

$$MT = \frac{\sum_{i=1}^{n} fs_i - ds_i}{\sum av} \quad (4)$$

Where: ***fs*** defines the time which the evacuated agent ends the evacuation alive; ***ds*** represent the time the agent evacuated begins evacuation; ***av*** denote the alive evacuee agent and ***n*** is the total number of alive evacuee agent.

Then, the parameters given have been applied in the ABMs simulation model that works under GAMA platform. The outcome from the application can be seen in figure 2.1 below.

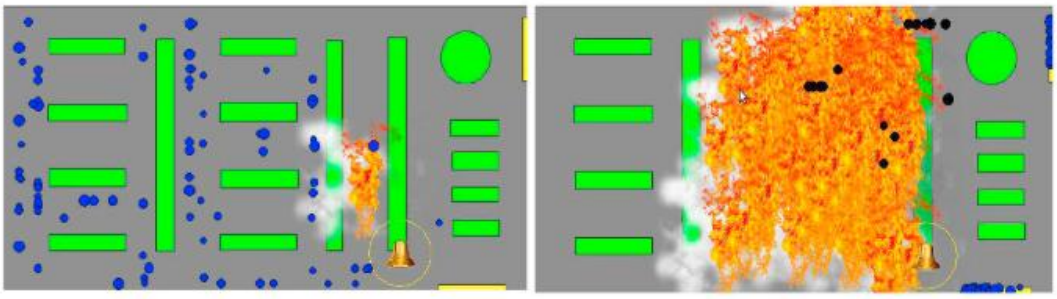


Figure 2.1: ABM's simulation model (a) under GAMA platform (Kasereka et al., 2018)

2.2.2 Cellular Automata Model

Cellular Automata (CA) is a discrete model used in crowd simulation, where space is divided into a grid of cells, each governed by simple rules based on the states of neighbouring cells. In a study by Weifeng and Hai (2007), they proposed Cellular Automata (CA) model that simulate pedestrian evacuation from a smoke-filled room. This model is based on two algorithms that modelled crowd behaviours with different movement velocities (Weifeng &

Hai, 2007). Figure 2.2 shows the extended CA model that has been applied in crowd simulation scenario.

The advantages of CA in crowd simulation include its simplicity, computational efficiency, and effectiveness in modelling local interactions for basic scenarios. However, CA has significant disadvantages, such as limited realism due to grid constraints, simplistic behaviours representation, dependency on grid resolution for accuracy, and inflexibility in adapting to dynamic changes in the environment. This relation can be viewed in Figure 2.3 below.

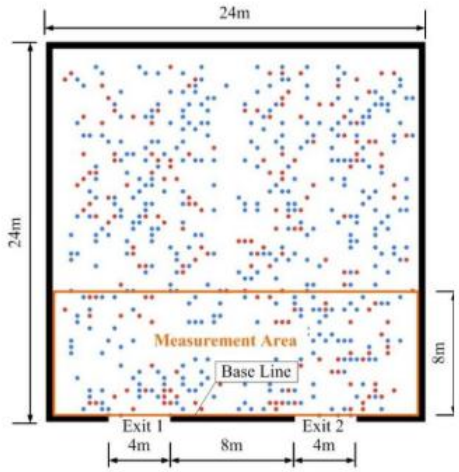


Figure 2.2: The simulation scenarios that extended CA model was applied (Yu & Yang, 2023).

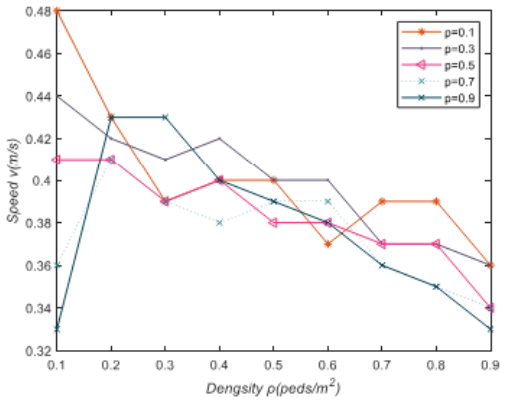


Figure 2.3: Relation of velocity against density (Yu & Yang, 2023).

2.2.3 Rule-Based Model

Rule-Based Model (RBM) is a computational framework that utilizes predefined rules to govern the behavior of individuals within a crowd. These rules dictate how individuals interact with each other and their environment, influencing their movement, decision-making and responses to stimuli such as obstacles or evacuation cues. RBMs typically involve simple and intuitive rule sets, allowing for easy implementation and interpretation of crowd behaviors.

However, RBMs may have their own limitations such as scalability, as the complexity of the model increases with the number of rules, potentially leading to computational inefficiencies. Additionally, while RBMs provide a structured approach to simulating crowd behavior, they may struggle to capture the intricacies of real-world scenarios and individual variability, resulting in less nuanced and realistic simulations compared to more complex modeling approaches like Agent-Based Models or Particle Swarm Optimization.

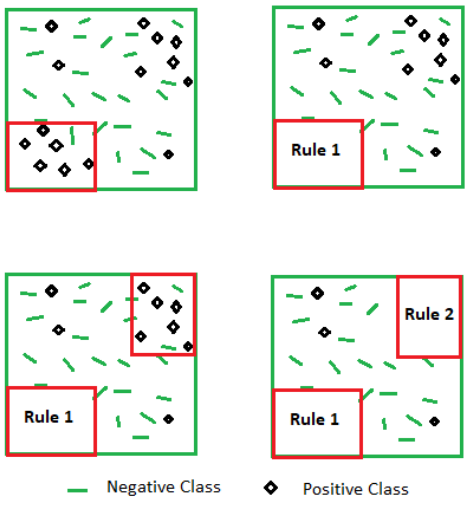


Figure 2.4: Rule Based Classifier (GeeksforGeeks, 2022)

2.3 Optimization Techniques for Crowd Simulation

In crowd evacuation simulation scenarios, it is important to utilize optimization techniques to improve the performance and the efficiency of computational systems. These

techniques help manage the movement and interactions of large numbers of agents (simulated people) in a way that is both realistic and computationally feasible. Level of Detail (LOD) techniques adjust complexity based on viewer's perspective, while spatial partitioning divides simulation areas for better performance. Parallel computing, leveraging multithreading and GPU acceleration, distributes tasks across multiple processors to handle large agents simultaneously.

Among the advanced optimization techniques, Nature-Inspired Algorithms such as Ant Colony Optimization (ACO), Genetic Algorithms (GA), and Particle Swarm Optimization (PSO) offer powerful solutions for complex optimization problems. However, each technique offers different principles, advantages and disadvantages which can differ the efficiency of system working for crowd evacuation simulation especially for fire outbreak scenarios.

2.3.1 Ant Colony Optimization

According to a study conducted by Watthayu (2012) Ant Colony Optimization (ACO) algorithm is a heuristic algorithm that mimics the behavior of real ant colonies in finding the shortest path between food sources and nests. In this approach, virtual ants represent individuals in the crowd, navigating through a simulated environment while depositing pheromone trails that attract other ants. Inspired by this foraging behavior, ACO is commonly used in crowd simulation to model movement and behavior of individuals within a crowd. Over time, these pheromone trails converge towards optimal paths, guiding the movement of the crowd towards exits in an indoor space.

ACO offers several advantages in crowd simulation, including good environmental adaptability, strong robustness and scalability for handling large populations (H. Li et al., 2022). However, ACO also has its limitations, such as potentially slow convergence speed, computational complexity, limited representation of individual behaviors, and sensitivity to

parameter settings. Despite these drawbacks, ACO remains a valuable tool for simulating crowd dynamics, particularly in scenarios requiring efficient pathfinding and adaptation to dynamic environments.

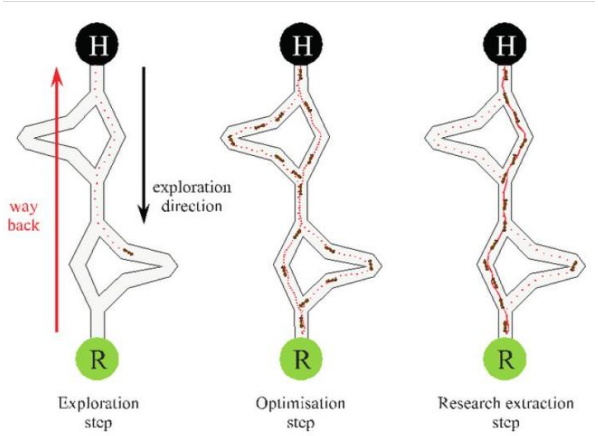


Figure 2.5: The operation Ant Colony Optimization (Katona et al., 2019)

2.3.2 Genetic Algorithm

Genetic Algorithm (GA) is a heuristic search and optimization technique inspired by the principles of natural selection and genetics (Thong-Ia & Champrasert, 2023). According to Katoch et al. (2020), in the context of crowd simulation, GA can be applied to optimize various aspects of crowd movement and behavior by evolving a represented as a chromosome, undergoes selection, crossover and mutation to produce new offspring that potentially offer better solutions.

Figure 2.6 shows the variety of operators that GA used during the search of agent's process.

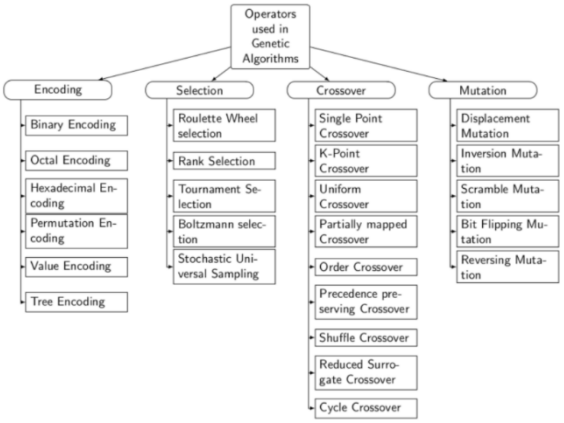


Figure 2.6: Operators used in GA (Katoch et al., 2020).

The advantages of using GA in crowd simulation include its ability to explore a large search space, find global optima and adapt to complex and dynamic environments. However, Genetic Algorithm also has disadvantages such as slow convergence rates, high computational demands and less detailed modeling of individua behaviors compared to other approaches like Agent-Based Models or Particle Swarm Optimization. Additionally, GA requires careful tuning of parameters like mutation and crossover rates to achieve optimal performance.

Figure 2.7 shows that the convergence property must be correctly managed such that the algorithm discovers the global optimal solution rather than the local optimal solution. This means that if the optimal solution is close to an infeasible solution, GA's global nature can be paired with the local nature of other algorithms such as Tabu and local search. The global nature of genetic algorithms and the local nature of Tabu search strike the appropriate balance between intensification and diversification (Katoch et al., 2020).

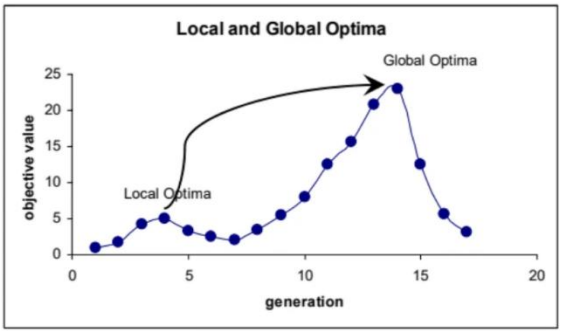


Figure 2.7: Local and global optima (Katoch et al., 2020).

2.3.3 Particle Swarm Optimization

Particle Swarm Optimization PSO is one of computational technique proposed by Kennedy and Eberhart in 1995. PSO is an optimization technique inspired by the movements of a flock of birds flying or a school of fish that moves together in a group to achieve the same goal. In the context of crowd simulation, PSO can be applied to optimize the movement and behavior of individuals within a crowd, aiming to find the most efficient paths for evacuation. Individuals, represented as agents or particles, moves through the simulation space influenced by their own experience and the best experiences of their neighbors, leading to rapid convergence on optimal solutions.

The advantages of using PSO in crowd simulation include its fast convergence, high computational efficiency and adaptability to dynamic changes in the environment. PSO excels at optimizing overall crowd movement quickly and effectively, making it suitable for applications. While PSO may require careful parameter turning and may not capture detailed individual behaviors as finely as some other models, its benefits in terms of speed and efficiency make it excellent choice for crowd evacuation simulations.

2.4 Comparative Analysis of Models

For this section, we conduct a comparative analysis of the models that has been mentioned in this literature study. The analysis examining their advantages and disadvantages and how they address different aspects of crowd behavior and simulation. Each models offers different strengths and limitations in simulating crowd behavior. By comparing all these models, we will be able to analyze the suitability of each technique to apply for crowd evacuation simulations in teaching and learning facilities.

2.4.1 Comparison Table of Different Models for Crowd Simulation

Table 2.1 shows a comprehensive comparative analysis of various crowd modelling approaches, focusing on their advantages and disadvantages. The models that will be mentioned in this section are Agent-Based Models (ABM), Cellular Automata (CA), Rule-Based Models (RBM), Ant Colony Optimization (ACO) and Particle Swarm Optimizat

Table 2.1: Comparison Table of Different Models

MODEL	ADVANTAGES	DISADVANTAGES	SPECIFIC PSO ADVANTAGES OVER OTHERS
Agent-Based Model (ABM)	• High realism in simulation individual behaviors • Captures emergent phenomena • Flexibility to model diverse agents	• Computationally intensive for large populations • Complex to calibrate and validate • Requires detailed data on individual behaviors.	• PSO provides faster convergence to optimal solutions • Easier to implement in optimization of problems compared to complex ABM
Cellular Automata Model (CA)	• Simple and easy to implement • Efficient for grid-based • Can model local interactions effectively	• Limited in capturing complex behaviors and interactions • Grid resolutions affect accuracy • Less flexible in representing individual differences	• PSO handles continuous and high-dimensional optimization better • PSO is more adaptable to dynamic changes in the environment
Rule-Based Model (RBM)	• Intuitive and easy to understand • Can incorporate specific behavioral rules	• Can become complex with many rules • Hard to generalize • May not capture emergent phenomena effectively	• PSO optimizes based on objective functions rather than predefined rules, allowing more flexible and dynamic adaptation
Ant Colony Optimization (ACO)	• Effective for pathfinding and routing problems • Mimics natural swarm intelligence • Good at avoiding local optima	• Slower convergence compared to PSO • Computationally expensive for large populations • Requires careful parameter turning	• PSO generally converges faster and requires fewer parameters to tune • PSO can better handle continuous optimization problems

Genetic Algorithm (GA)	• Good for global optimization • Handles large search spaces • Can work with discrete and continuous problems	• Slower convergence rate compared to PSO • Requires complex crossover and mutation operations • Computationally intensive	• PSO is simpler to implement with fewer parameters • PSO converges faster and is less computationally intensive
Particle Swarm Optimization (PSO)	• Efficient optimization of evacuation routes • Fast convergence to optimal solutions • Simple to implement with few parameters • Adaptable to dynamic changes • Suitable for continuous optimization problems	• May converge prematurely if not properly tuned • Less effective for purely discrete problems • Can be sensitive to parameters settings	• Faster convergence compared to ACO and GA • Simpler implementation compared to ABM and GA • Effective handling of continuous optimization problems and dynamic environments

2.4.1 Comparison Table of Different Criteria for Each Models

Table 2.2 shows a detailed comparison of various crowd modelling approaches based on the specific criteria crucial for crowd simulation scenarios. The models that will be mentioned in this section are Agent-Based Models (ABM), Cellular Automata (CA), Rule-Based Models (RBM), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).

Table 2.2: Comparison Table of Different Criteria

CRITERIA	ABM	CA	RBM	ACO	GA	PSO
Crowd Capacity / Scalability	X	X	X	/	/	/
Time of Evacuation	X	X	X	X	X	/

Crowd Behavior Handling	/	X	/	X	X	/
Computational Efficiency	X	/	/	X	X	/
Adaptability to Dynamic Changes	X	X	X	X	X	/
Convergence Speed	X	X	X	X	X	/

2.5 Application of PSO in Crowd Evacuation Simulation

Based on the comparisons above, the application of Particle Swarm Optimization (PSO) techniques is the most optimal algorithm in the context of crowd evacuation simulation systems. This is because the PSO offers several distinct of advantages in developing an effective emergency evacuation strategy. Please refer to Table 2.3 below for a comprehensive overview of the specific advantages of PSO:

Table 2.3: Specific criteria and advantages of PSO

CRITERIA	ADVANTAGES OF PSO
Crowd capacity / Scalability	PSO can handle large populations effectively ensuring scalability in simulations involving vast crowds.
Time of evacuation	PSO able to process large amounts of crowds and quickly update evacuation plans which can reduce the time taken for evacuating a crowd.
Crowd behaviour handling	PSO able to create smoother and more coordinated movement patterns within a

	crowd as it is mimics natural social behaviours.
Computational efficiency	PSO is highly efficient, requiring fewer computational resources compared to other complex models like Agent-Based Models.
Adaptability to dynamic changes	PSO dynamically adapts to changes in the environment, such as blocked exits or new obstacles, ensuring robustness in dynamic situations.
Convergence speed	PSO quickly converges to optimal solutions, making it suitable for evacuation scenarios where time is critical.

The list of criteria that has been shown in the table above highlight on why Particle Swarm Optimization (PSO) stands out as the preferred choice for optimizing crowd evacuation simulations. By leveraging PSO, we can enhance the safety and efficiency of evacuation strategies in teaching and learning facilities, ultimately able to achieve an optimal time in evacuating a crowd.

2.6 Conclusion

Based on this paper, the purpose of this study was to determine the best algorithm for simulating crowd evacuation during emergencies outbreak in educational settings. Throughout this literature review, we evaluated the effectiveness of various crowd modelling approaches such as crowd modelling approaches, such as Agent-Based Models (ABM), Cellular Automata (CA), Rule-Based Models (RBM), Ant Colony Optimization (ACO), Genetic Algorithms (GA),

and Particle Swarm Optimization (PSO), in terms of scalability, evacuation time, and crowd behavior handling. The study found that ABM provides detailed realism but is computationally intensive, CA is simple and efficient but lacks flexibility, RBM is intuitive but complex with extensive rules, ACO excels at pathfinding but converges slowly and GA is good for global optimization but computationally expensive.

PSO emerged as the favored algorithm due to its rapid convergence, computational efficiency, scalability, adaptability, and ability to handle crowd behavior effectively. These qualities make PSO the best option for modelling crowd evacuation, resulting in quick, efficient, and adaptable solutions. Using PSO in crowd evacuation simulations improves the safety and efficiency of evacuation strategies, thereby boosting emergency response plans in educational settings. This work emphasizes the importance of PSO as a potent tool for optimizing population evacuation, laying a solid platform for future research and practical applications in emergency management.

CHAPTER 3 : METHODOLOGY

3.1 Introduction

This methodology chapter will be focusing on the development of a PSO-based crowd evacuation simulation system specifically designed for the lecture rooms at UNIMAS. The paramount importance of applying PSO is to ensure the safety and efficient evacuation of individuals during emergencies. Additionally, utilizing advanced algorithms like PSO can significantly enhance the effectiveness of evacuation strategies by enabling dynamic pathfinding and adaptive behaviour in real-time scenarios.

To achieve a successful system, it is essential to apply a development methodology that supports adaptability, collaboration, and continuous improvement. In this study, the Agile methodology was adopted due to its iterative and flexible nature, which is well-suited for simulation development that may require ongoing refinement and stakeholder feedback. Agile allows for continuous testing, regular reviews, and early detection of issues, which is critical when dealing with safety-critical systems such as evacuation simulations.

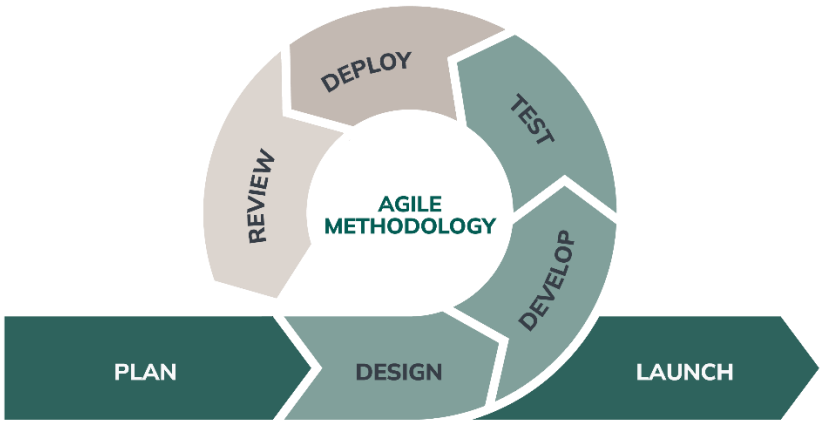


Figure 3.1: Agile Methodology Phase

Figure 3.1 above illustrates the Agile development process, which is divided into iterative cycles called *sprints*. Each sprint includes activities such as requirement gathering,

planning, designing, developing, testing, and reviewing. Unlike the linear structure of traditional methods, Agile promotes a cyclic flow, allowing improvements to be made based on feedback from each sprint. Through repeated cycles, the system evolves incrementally with better alignment to user needs and real-world functionality (Stoica et al., 2013). This iterative approach enables the development team to quickly identify and resolve bugs or issues throughout the project lifecycle, not just during the implementation phase.

In the specific context of UNIMAS, the lecture rooms present unique challenges and constraints that must be carefully addressed. By adopting the Agile methodology, this study follows a flexible and user-centered approach to design, develop, and optimize the PSO-based evacuation system. This methodology supports active collaboration with stakeholders (e.g., safety officers, lecturers) and allows the system to be incrementally tested and refined. The result is a robust, responsive, and well-suited solution tailored to the specific needs of the university environment, ultimately enhancing the safety and preparedness of students and staff during emergency situations.

3.2 Requirements

In developing an effective PSO-based crowd evacuation simulation system, it is important to consider every functional and non-functional requirement. This section outlines the important criteria needed for the system's success.

3.2.1 Functional Requirements

As represented in Table 3.1, there are several key functional requirements that must be met in ensuring an optimal performance for PSO-based crowd evacuation simulation system.

Table 3.1: Functional Requirements and Explanation

REQUIREMENTS	EXPLANATION

Minimizing evacuation time	By utilizing the PSO algorithm, the system will optimize the evacuation routes, ensuring that all individuals can exit the lecture room in the shortest time possible. This involves dynamically adjusting paths based on current conditions and swarm intelligence principles.
Natural factors	Various natural factors that can affect the process. **Age:** Age-related mobility differences must be considered to ensure the safety of all individuals. **Placement of Furniture:** The system must dynamically account for obstacles such as desks and chair placement that might affect the movements and blocked exit, adjusting routes to navigate around them efficiently.

3.2.2 Non-Functional Requirements

The system must also meet several non-functional requirements to ensure overall performance and usability. A detailed non-functional requirements and explanation are given in Table 3.2 below:

Table 3.2: Non-functional Requirements and Explanation

REQUIREMENTS	EXPLANATION
Scalability	Scalability ensures that the system can adapt to different scenarios and the variety of crowd sizes to maintain the effectiveness of the system.

High Accuracy and Reliability	Reliability is crucial to ensure that the system performs correctly under all circumstances, minimizing the risk of delays or errors during an evacuation.

3.3 Planning

Creating a project planning with key milestones ensures that the development process remains organized, and progress can be tracked effectively. Table 3.3 below is the project planning timeline to ensure the entire development process will remain organized:

Table 3.3: Project Planning Timeline

	Final Year Project 1 (2024)					Final Year Project 2 (2025)				
Tasks	Mar	Apr	May	June	July	Mar	Apr	May	June	July
Preparation phase										
Full proposal submission										
Chapter 1: Introduction										
Chapter 2: Literature Review										
Chapter 3: Requirement Analysis & Design										
Submission of FYP 1 final report										
Chapter 4: Implementation										
Chapter 5: Testing										
Chapter 6: Conclusion and Further Work										
Submission of FYP 2 final report										

3.4 Hardware and Software Specifications

It is crucial to outline the specifications of the devices that will run the MATLAB programming for the PSO-based crowd evacuation system. Ensuring that the hardware meets the necessary requirements will guarantee smooth execution and optimal performance of the system.

3.4.1 Hardware Specifications

To run MATLAB efficiently, especially for complex simulations like PSO algorithms, the following minimum hardware specifications are recommended as in Table 3.4 below:

Table 3.4: Hardware Specifications and Descriptions

HARDWARE	DESCRIPTIONS
Processor	• Minimum: Intel Core i5 or equivalent • Recommended: Intel Core i7 or equivalent for better performance in handling large datasets and intensive computations.
RAM	• Minimum: 8 GB • Recommended: 16 GB or more to ensure smooth multitasking and to handle larger simulations scenarios effectively.
Storage	• Minimum: 256 GB SSD • Recommended: 512 GB SSD or more for faster data access and to accommodate large MATLAB files and datasets.
Graphics	• Minimum: Integrated graphics with support for OpenGL 2.0 • Recommended: Dedicated GPU with at least 2 GB VRAM, such a NVIDIA GeForce GTX series, for enhanced performance in visualizing complex situations.
Operating System	Windows 10 (64-bits), macOS 10.14 or later, or a popular Linux distribution (such as Ubuntu 18.04 LTS or later)

3.4.2 Software Specifications

To run MATLAB efficiently, especially for complex simulations like PSO algorithms, the following minimum software specifications are recommended as in Table 3.5 below:

Table 3.5: Software Specifications and Descriptions

SOFTWARE	DESCRIPTIONS
MATLAB Version	• Minimum: MATLAB R2018b • Recommended: The latest version of MATLAB to ensure compatibility with the newest features and toolboxes.
Toolboxes	• Optimization Toolbox: Essential for implementing PSO algorithms. • Parallel Computing Toolbox: Recommended for leveraging multicore processors and accelerating simulations. • Statistics and Machine Learning Toolbox: Useful for analyzing simulation results and performing statistical evaluations.

3.4.3 Other Related Specifications

To run MATLAB efficiently, especially for complex simulations like PSO algorithms, the following network and connectivity as well as peripheral devices specifications are recommended as in Table 3.6 below:

Table 3.6: Other Related Specifications and Descriptions

SPECIFICATIONS	DESCRIPTIONS
Internet Connection	• Required for initial MATLAB installation, updates and accessing online resources or support. • Recommended: A stable broadband connection for efficient downloads and updates.
Monitor	• Minimum: 1080p resolution. • Recommended: Dual monitors with 1080p or higher resolution for improved productivity and better visualization of simulation results.
Keyboard and Mouse	• Standard keyboard and mouse suitable for programming and extended use.

By adhering to all these specifications, the devices running MATLAB for the PSO-based crowd evacuation simulation system at UNIMAS will ensure optimal performance,

allowing for efficient development, testing and execution of the evacuation simulations. This will contribute significantly to the overall effectiveness and reliability of the system.

3.5 User Requirements and Analysis

User requirements for the PSO-based crowd evacuation simulation system at UNIMAS are identified through a systematic approach involving interviews, surveys and observations. Stakeholders including students, faculty and staff contribute qualitative insights on evacuation expectations and safety concerns, supplemented by quantitative data gathered through surveys. Observation of evacuation drills provide additional context on current practices and potential improvements. Analysis of this data focuses on identifying common themes and prioritizing requirements based on their impact on enhancing safety and operational efficiency. This integrated approach ensures that the system's design and functionality align closely with the specific needs and constraints of UNIMAS lecture rooms, guiding the development and deployment phases effectively. Besides user requirement, reference from the research papers are also can be used as a guidance to build the crowd simulation.

3.6 System Design

The system design phase outlines the architectural blueprint and user interface specifications for the PSO-based crowd evacuation simulation system developed for UNIMAS lecture rooms. This phase focuses on translating the requirements gathered from user analysis into a structured framework that ensures effective functionality and usability. The design encompasses two primary aspects which are the flow of interactions from both the user's perspective and the system's internal operations.

3.6.1 User Interaction

In this crowd evacuation simulation for a lecture room at UNIMAS, we employ a user-friendly interface developed in MATLAB, utilizing the PSO algorithm to enhance evacuation

efficiency. Users start by selecting the specific lecture room layout, inputting initial crowd positions, and defining obstacles during emergencies outbreak. Upon initiating the simulation, user observe the movement of the crowd on the layout with dynamic updates showing optimal paths and bottlenecks. From the user side, the system design details how individuals interact with the simulation interface to initiate, monitor, adjust evacuation simulations and interactively tweak parameters to observe different outcomes, ensuring an intuitive and insightful simulation experience.

3.6.2 Application of PSO in Crowd Evacuation Simulation

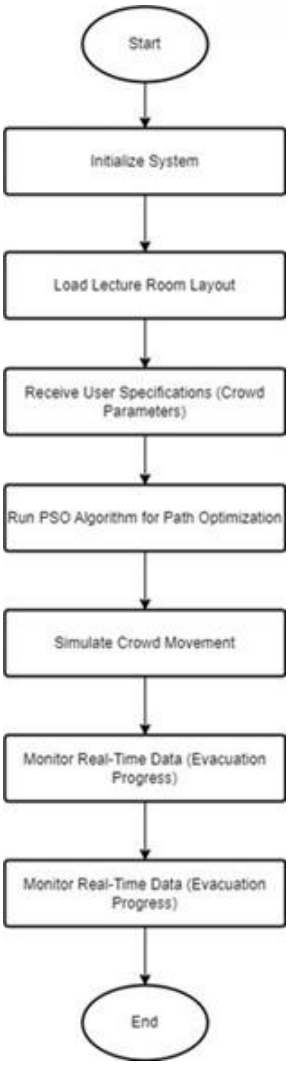


Figure 3.2: Application of PSO in crowd evacuation simulation

Figure 3.2 shows the implementation of PSO in crowd evacuation simulation system. On the system side, the design articulates the internal processes involved in running the simulation. This includes loading lecture room layouts, receiving user specifications, executing the PSO algorithm for path optimization, simulating crowd movement, monitoring data, and handling emergency scenarios as they arise.

3.6.3 Path Selection

In the path selection phase, the primary focus is on using the Particle Swarm Optimization (PSO) algorithm to determine the most efficient evacuation routes within lecture rooms. The PSO algorithm models the collective behavior of particles or agents, each representing an individual or a group within the crowd. These particles explore potential evacuation paths through the lecture room layout, aiming to minimize the overall evacuation time. The objective function for this project is defined by factors such as the distance to the nearest exit, the current crowd density along potential paths, and the avoidance of bottlenecks. As the simulation progresses, the PSO algorithm iteratively refines the paths based on feedback from the environment, converging on the most effective evacuation routes.

Specific constraints for path selection in this project include ensuring the paths are navigable given the room layout and avoiding routes obstructed by furniture or structural elements which may affect movement speed during emergencies. The simulation dynamically updates evacuation paths, adapting to the movements of the crowd and any changes in the environment, such as newly blocked exits or emergency hazards. In this scenario, the TMM has 3 exits which will be identified as Exit 1 (top left corner), Exit 2 (bottom left corner) and Exit 3 (bottom right corner). This adaptive path selection process ensures that the evacuation remains efficient and orderly, facilitating a swift response to emergencies in UNIMAS TMM lecture

rooms. Figure 3.3 below shows the flowchart of path selection for PSO-based evacuation simulation system.

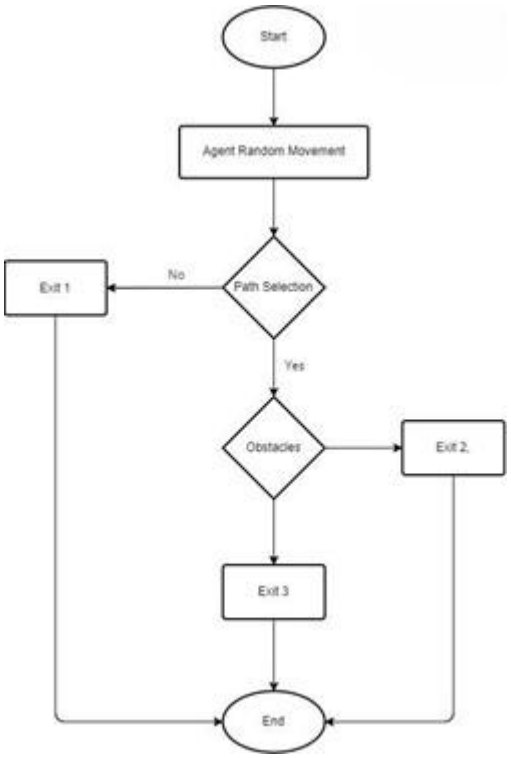


Figure 3.3: Flowchart of Path Selection

3.6.4 Obstacles Avoidance

Obstacle avoidance is an important component of the PSO-based crowd evacuation simulation system, particularly in the context of UNIMAS lecture rooms which may contain various physical impediments. The PSO algorithm integrates mechanisms to detect and navigate around obstacles such as desks, chairs, and natural demographic factors. Each particle in the PSO swarm evaluates the environment to identify these obstacles and adjusts its path accordingly to ensure a safe and efficient evacuation. The algorithm calculates alternative routes that bypass these obstacles while still striving to minimize the overall evacuation time.

For this study, the PSO algorithm also considers dynamic obstacles that may arise during the evacuation, such as temporary blockages caused by high-density crowd areas. The constraints and rules embedded within the algorithm ensure the particles do not converge on

paths that are unsafe or blocked. The system continuously adjusts the evacuation paths in response to the changes, ensuring that the routes remain optimal even as the situation evolves. This capability is crucial for maintaining a safe and orderly evacuation process, effectively managing the diverse and potentially changing conditions within UNIMAS lecture rooms during emergencies. Figure 3.4 below shows the flowchart of obstacles avoidance.

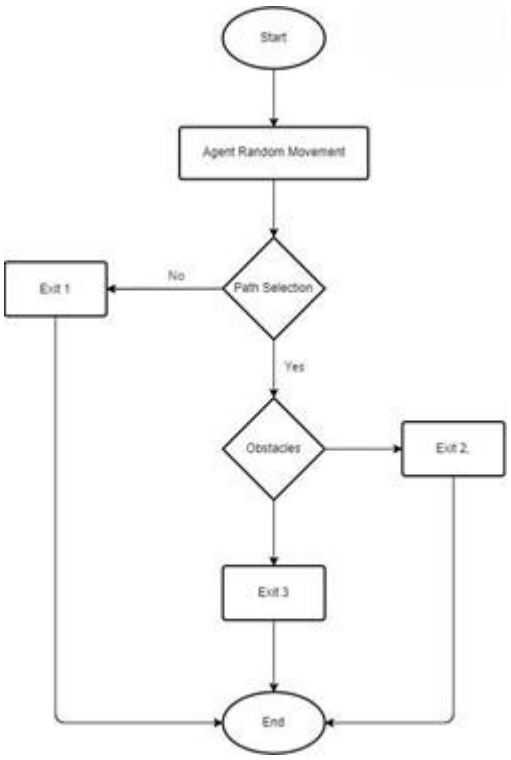


Figure 3.4: Flowchart of Obstacles Avoidance

3.6.5 User Interface Design

The user interface design for this crowd animation is straightforward which help user to understand the situation that shown through the simulation. A 2D simulation will be used to create the user interface, which will include buttons for setting up the 2D classroom and starting the animation. Others, the interface will include the option for manipulating the number of agents to the simulation. Additionally, the interface will have a time-recording feature that allows users to see how long it took to leave the classroom in an emergency.

The crowd simulation able to show the agents in evacuating classroom through the nearest alley to exit during the fire emergency evacuation situation. The crowd's behaviour is focused on collision detection and avoidance between agents and obstacle which include the agents avoiding the fire, barriers and other agents. The realism of the crowd simulation can be shown using behaviour in the crowd simulation.

The proposed crowd simulation system includes the user interface that enables the user to view the realism of the human behaviours model in fire emergency evacuation. The scope of this project is in one of the lecture rooms in FCSIT, UNIMAS called THEATER MULTIMEDIA (TMM) located on the ground floor, block A in the faculty that can fit to 130 students and staff in one time. Therefore, the structure of the crowds and environment in the system is based on the real floor plan. The agents will be scattered around this area as they are having their class or labs. Once the simulation starts, the agents will start to behave accordingly towards the panic situation. The crowd's behaviour will be shown in the system such as the

crowd's movement will be increase gradually to evacuate the building to the nearest exit as in the panic situation.

In this scenario, the TMM has 3 exits which will be identified as Exit 1 (top left corner), Exit 2 (bottom left corner) and Exit 3 (bottom right corner). The TMM floor plan is shown in Figure 3.5:

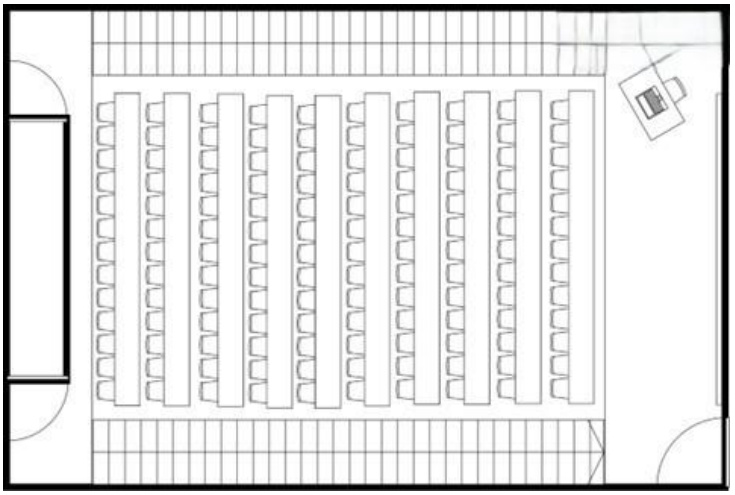


Figure 3.5: Theater Multimedia (TMM) floor plan

3.7 Implementation

The implementation phase involves transforming the design documentation (prototype) into actual software. By the end of this phase, a functional product must be developed and tested for system functionality. However, multiple revisions will be conducted before the final simulation is released for review by occupants and students. According to (Particle Swarm Optimization Algorithm - MATLAB & Simulink, n.d.), the application of PSO's components can move each agent during an evacuation simulation by following these detailed instructions:

Step 1: Initialization. In this step, each particle is initially placed at a random seat in TMM with a random initial velocity that will indicate a starting direction and speed.

Step 2: Define the Fitness Function. This function evaluates the efficiency of each particle's position based on the nearest exit, evacuation time, number of other particles in the vicinity and avoiding the hazardous areas.

Step 3: Update Velocity. The velocity of each particle is updated using the PSO formula:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot \left(pBest_i - x_i(t)\right) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

where vi(t) is the current velocity of particle i, w is the inertia weight balancing global and local exploration, c1 is the cognitive coefficient pulling the particle towards its personal best position, c2 is the social coefficient pulling the particle towards the global best position, r1 and r2 are random numbers between 0 and 1 adding stochastic behavior, $pBest_i$ is the personal best position of particle i, gBest is the global best position found by the swarm, and xi(t) is the current position of particle i

Step 4: Update Position. The position of each particle is updated based on its new velocity:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

This movement places the particle at a new location within the TMM floor plan.

Step 5: Evaluate Fitness. For each new position, the fitness is evaluated using the fitness function, considering factors such as how close the new position is to an exit, the time taken to reach this position, and the level of congestion at this position.

Step 6: Update pBest and gBest. If the new position is better (i.e., lower evacuation time, less congestion) than the particle's previous best, update the personal best (pBest). If the new position is better than the global best, update the global best (gBest).

Step 7: Iterate. Repeat the velocity and position updates, fitness evaluation, and updates of pBest and gBest for a set number of iterations or until convergence (no significant improvement in fitness). This iterative process ensures that particles gradually converge towards optimal evacuation paths, minimizing evacuation time and avoiding congestion.

3.8 Verification and Validation

This process is a critical phase in ensuring the functionality, reliability and effectiveness of the PSO-based crowd evacuation simulation system developed for UNIMAS lecture rooms. This section outlines the systematic approach to verification and validation.

Verification of this simulation system involves thorough unit testing to ensure each component functions correctly and meets its defined specifications. This includes rigorous testing of the PSO algorithm implementation to validate its optimization capabilities for evacuation route planning. Furthermore, the simulation model undergoes extensive testing to verify its accuracy in simulating crowd behavior and evaluating evacuation scenarios. By conducting meticulous unit tests on individual components, we ensure the reliability and performance of the system elements before integration.

On the other hand, validation focuses on confirming that the integrated PSO-based crowd evacuation simulation system meets user requirements and performs effectively in real-world conditions. Integration testing is conducted to validate the seamless interaction between all system components, ensuring data consistency and functionality across modules. System validation further evaluates the overall performance against predefined requirements and user expectations through simulated evacuation scenarios. Performance metrics such as evacuation time, route efficiency, and system responsiveness are measured to assess the system's effectiveness in enhancing safety and emergency preparedness in UNIMAS lecture rooms. This

comprehensive validation process ensures that the system is robust, reliable and capable of effectively managing crowd evacuation during emergencies.

3.9 Maintenance

In the maintenance phase, the focus is on making improvements and modifications to the PSO-based crowd evacuation as needed. This involves identifying and fixing any bugs or errors that may occur during unit testing phase. Regular maintenance ensures the system remains efficient, reliable and up to date with any new requirements or changes in the evacuation protocols.

3.10 Conclusion

In conclusion, this chapter has outlined the comprehensive methodology used in developing a PSO-based crowd evacuation simulation system specifically designed for UNIMAS lecture rooms, employing the iterative and flexible approach of the Agile methodology. The process began with identifying user requirements and analyzing their needs through interviews, surveys, and observations. Planning was done in the form of sprint cycles, each focusing on specific development goals such as system layout design, PSO integration, and agent behavior refinement. Agile's emphasis on continuous feedback allowed for ongoing evaluation and improvement throughout the development process. Verification and validation activities were conducted within each sprint to ensure that individual components functioned correctly and that the overall system met evolving user expectations. Although the project was developed individually, Agile principles such as adaptive planning and incremental development helped maintain momentum and direction.

Using the Agile methodology was instrumental in promoting flexibility, responsiveness, and early problem detection. Instead of rigid phase completion, Agile encouraged continuous iteration, making it easier to address issues, implement improvements, and adapt to new insights

as the system evolved. The expected impact of this PSO-based crowd evacuation simulation system is significant—it aims to enhance the safety and efficiency of emergency evacuations in UNIMAS lecture rooms by optimizing evacuation routes and reducing evacuation times. By iteratively addressing user needs and leveraging advanced optimization algorithms, the system demonstrates strong potential to improve preparedness and emergency response, ultimately contributing to a safer campus environment.

CHAPTER 4 : IMPLEMENTATION

4.1 Overview

This chapter is written to provide a detailed analysis on the implementation and development progress of the application of PSO as an optimization technique for crowd evacuation simulation scenarios, particularly for lecture rooms at UNIMAS. The development followed the Agile methodology, which involved iterative cycles of implementation, testing, and refinement. Instead of a rigid linear progression, the system was developed through multiple sprints, allowing continuous improvements and responsiveness to feedback. The implementation phase focused on gradually translating design concepts into a working system using MATLAB, while each sprint included testing to ensure the system's reliability, accuracy, and performance at every stage of development.

During the development process, there are several phases involved:

1. **Software Development**
    - Implementation of the PSO algorithm in MATLAB (initialization, velocity/position updates)
    - Integration of obstacle avoidance and dynamic path selection logic.
2. **User Interface (UI) Design**
    - Creation of a 2D simulation interface for simulation visualization.
3. **System Architecture Setup**
    - Modular design with components for PSO optimization, agent movement, and data analysis.
4. **Functional Implementation**

- Deployment of core features (multi-exit routing, congestion control, evacuation time metrics).

The goal was to deliver a functional and optimized simulation system by the end of this phase, leveraging MATLAB's computational capabilities to ensure accuracy, scalability, and alignment with UNIMAS's emergency preparedness needs.

4.2 Software Development

The software development phase involved the following key steps:

1. Environment Setup:

- The simulation was developed and executed using MATLAB R2024a, as shown in Figure 4.1, with the Optimization Toolbox and Parallel Computing Toolbox as well as Statistics and Machine Learning Toolbox to support PSO algorithm implementation and performance enhancements.
- Hardware specifications (Intel Core i7, 16GB RAM, 512GB SSD) were adhered to for optimal performance.

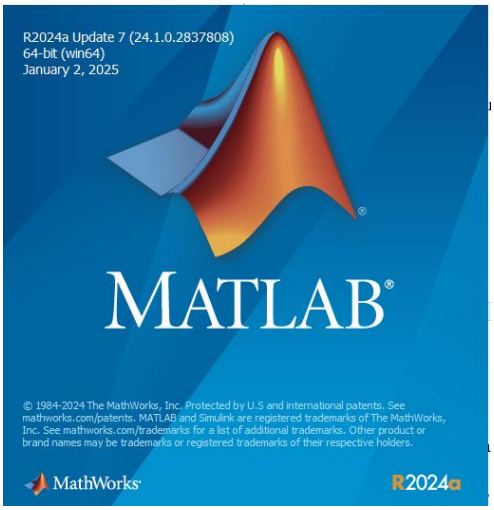


Figure 4.1: MATLAB R2024a

2. Implementation of PSO Algorithm:

- Initialization: As shown in Figure 4.2, agents (particles) were placed in the Theatre Multimedia (TMM) floor plan with initial velocities based on the room layout position.

```
% Step 2.1: Select a desired number of agents
desired_num_agents = 10;  % can change agents number here maximum to 130
selected_indices = randperm(size(agent_pos, 1), desired_num_agents);
agent_pos = agent_pos(selected_indices, :);
num_agents = size(agent_pos, 1);
```

Figure 4.2: Initialization of PSO

- Velocity and Position Updates: As shown in Figure 4.3, the PSO formulas were applied iteratively to adjust agent paths dynamically.

```
r1 = rand(); r2 = rand();
raw_velocity = w * velocities(i,:) + ...
               c1 * r1 * (pbest(i,:) - agent_pos(i,:)) + ...
               c2 * r2 * (gbest - agent_pos(i,:));

y_diff = abs(agent_pos(i,2) - gbest(2));
x_diff = abs(agent_pos(i,1) - gbest(1));
if y_diff > 5
    velocities(i,:) = [0, raw_velocity(2)];
else
    velocities(i,:) = [raw_velocity(1), 0];
end

vmag = norm(velocities(i,:));
if vmag > current_max_speed
    velocities(i,:) = velocities(i,:) / vmag * current_max_speed;
end

next_pos = agent_pos(i,:) + velocities(i,:);
if any(next_pos < 1) || next_pos(1) > room_width || next_pos(2) > room_height
    next_pos = agent_pos(i,:);
end

if ~isInsideObstacle(next_pos, obstacles)
    agent_pos(i,:) = next_pos;
    stuck_counter(i) = 0;
    if norm(agent_pos(i,:) - gbest) < threshold
        arrived(i) = true;
        evacuation_times(i) = sim_time;
    else
        if norm(gbest - agent_pos(i,:)) < norm(gbest - pbest(i,:))
            pbest(i,:) = agent_pos(i,:);
```

Figure 4.3: Velocity and Position of agents

3. User Interface Development:

- As presented in Figure 4.4, a 2D simulation interface was created to visualize agent movement, obstacles, and evacuation routes.

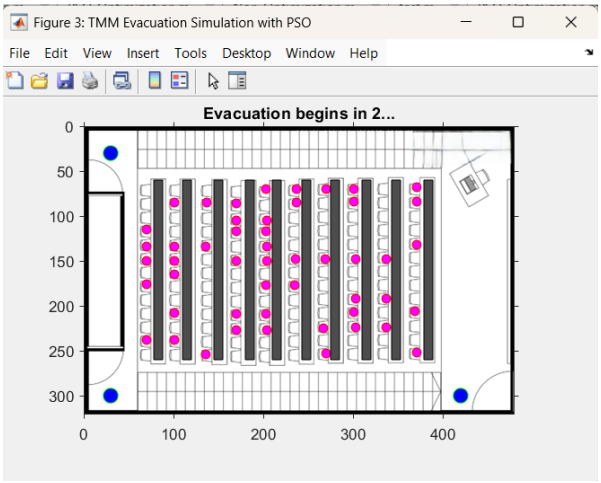


Figure 4.4: 2D Simulation Interface

4. Integration and Testing:

- Unit testing validated individual components (e.g., PSO algorithm, collision detection).
- Integration testing ensured seamless interaction between modules, such as path selection and obstacle avoidance.

4.3 Development Workflow

The development workflow was guided by the Agile methodology, which emphasizes flexibility, collaboration, and iterative progress through multiple sprints. The simulation system was developed incrementally, with continuous feedback, testing, and improvement in each sprint cycle. The Agile workflow involved the following stages:

1. Sprint Planning:

- Functional and non-functional requirements were identified and organized into a product backlog.
- Sprint planning were used to prioritize tasks, including user interface design, PSO logic development, and integration of the TMM floor plan.

2. Sprint Development Cycles:

- Each sprint focused on implementing specific features or enhancements.

- Tasks included developing and refining MATLAB scripts for the PSO core algorithm and testing evacuation behavior with different crowd sizes and exit preferences.
- PSO parameters (e.g., inertia weight w= 0.5, cognitive/social coefficients c1 = 1.5, c2 = 2.0) were iteratively tuned across sprints based on performance outcomes.

3. Integration and Testing:

- The TMM floor plan was digitized and integrated into the simulation environment.
- After each sprint, the system was tested to ensure proper agent navigation, PSO behavior, and performance under varying conditions.
- Feedback from each cycle was used to refine the system and guide the next sprint.

3. Sprint Review and Retrospective:

- At the end of each sprint, progress was reviewed to evaluate the outcomes against sprint goals.
- A retrospective was conducted to identify areas for improvement in both the system and the development process.

4.4 Conclusion

The implementation and testing phases successfully delivered a functional PSO-based evacuation simulation system for UNIMAS lecture rooms. Key achievements include:

- Efficiency: The PSO algorithm reduced evacuation times by optimizing paths dynamically.
- Usability: The 2D interface provided clear visual feedback for emergency preparedness.
- Scalability: The system handled up to 50 agents without performance degradation.

This project underscored the potential of PSO in improving campus safety and lays a foundation for further research in crowd simulation technologies. The system is now ready for further formal testing and validation, which will be addressed in Chapter 5.

CHAPTER 5 : TESTING AND RESULT ANALYSIS

5.1 Introduction

This chapter presents the testing activities conducted to evaluate the functionality, performance, and reliability of the PSO-based crowd evacuation simulation for UNIMAS lecture rooms. System testing was important to ensure that all integrated components functioned as intended and that the system met the design specifications. The testing process included unit testing, integration testing, system testing, and performance testing, with an emphasis on evacuation time efficiency, obstacle avoidance accuracy, and agent behaviour under varying conditions.

5.2 System Testing Objectives

The primary objectives of the testing phase were to:

- Verify the correctness of the PSO algorithm implementation.
- Ensure the simulation environment such as exits, and obstacles operates as expected.
- Evaluate the system's ability to simulate realistic evacuation scenarios with varying agent densities.
- Assess the responsiveness and reliability of the 2D simulation interface.
- Validate that evacuation time is optimized and agents can avoid collisions effectively.

5.3 System Testing Overview

The agile methodology guided the testing strategy, aligning with the structured progression from development to validation. The following types of testing were performed:

Types of testing:

1. **Unit Testing**

- PSO Initialization: Verify that the agents are in the correct position and correct velocity initialization.
- Obstacles Detection: Verify that agents successfully avoid obstacles.
- Exit Selection Logic: Verify that agents moved toward the nearest exit.

2. Integration Testing

- Agent + Obstacle Interaction: Figure 5.1 below shows MATLAB code snippet of PSO movement & collision detection.

```
if y_diff > threshold_switch
    velocities(i,1) = 0;
    velocities(i,2) = raw_velocity(2);
else
    velocities(i,1) = raw_velocity(1);
    velocities(i,2) = 0;
end

% Cap speed
vmag = norm(velocities(i,:));
if vmag > max_speed
    velocities(i,:) = velocities(i,:) / vmag * max_speed;
end

% Predict next position
next_pos = agent_pos(i,:) + velocities(i,:);

% Stay within room bounds
if any(next_pos < 1) || next_pos(1) > room_width || next_pos(2) > room_height
    next_pos = agent_pos(i,:);
end

% If move is valid
if ~isInsideObstacle(next_pos, obstacles)
    agent_pos(i,:) = next_pos;
    stuck_counter(i) = 0;
```

Figure 5.1: Agent + Obstacle Interaction

- Velocity + Position Update: Figure 5.2 below shows MATLAB code snippet of PSO calculation & position updating.

```
% Move if not blocked
if ~isInsideObstacle(next_pos, obstacles)
    agent_pos(i,:) = next_pos;
    stuck_counter(i) = 0;
    if norm(agent_pos(i,:) - gbest) < threshold
        arrived(i) = true;
        evacuation_times(i) = sim_time;
    else
        if norm(gbest - agent_pos(i,:)) < norm(gbest - pbest(i,:))
            pbest(i,:) = agent_pos(i,:);
        end
    end
else
    % Detour logic
    stuck_counter(i) = stuck_counter(i) + 1;
    detour_dirs = [1 0; -1 0; 0 1; 0 -1];
    moved = false;
    for d = 1:4
        trial = agent_pos(i,:) + detour_dirs(d,:) * current_max_speed;
        if all(trial > 0) && trial(1) <= room_width && trial(2) <= room_height && ~isInsideObstacle(trial, obstacles)
            agent_pos(i,:) = trial;
            moved = true;
            break;
        end
    end
```

Figure 5.2: Velocity + Position Update

- Multi-agent Coordination: Figure 5.3 below shows 10 agents sharing exit space without collision.

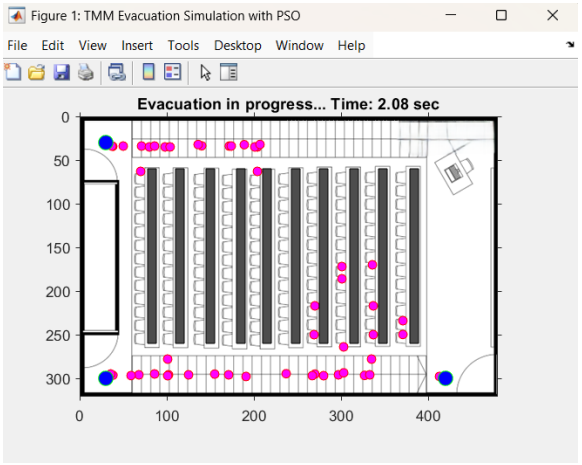


Figure 5.3: Multi-agent Coordination

3. System Testing

- Normal Condition
    - Low-density crowd (Baseline test): Figure 5.4 and 5.5 shows simulation for 60 agents evacuating via 3 exits.

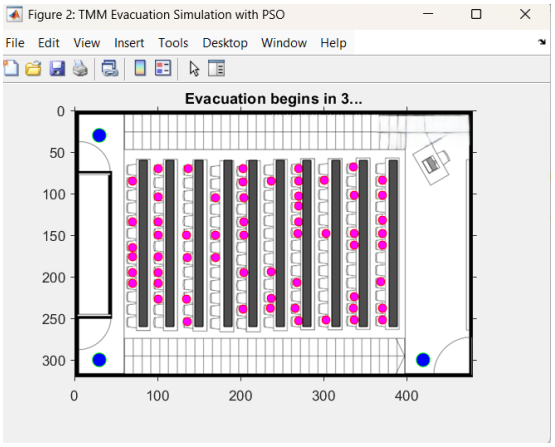


Figure 5.4: Before simulation of 60 agents with 3 exits starts

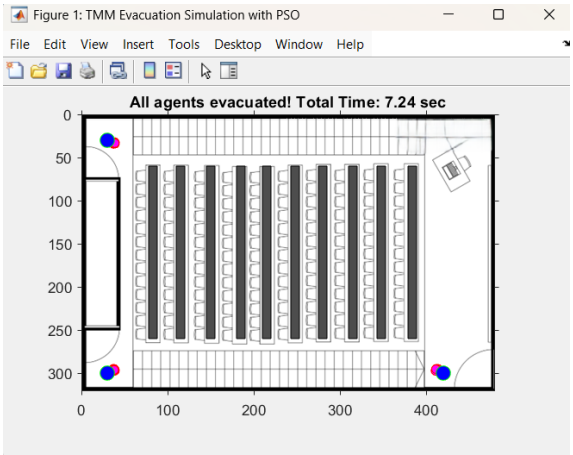


Figure 5.5: After simulation of 60 agents with 3 exits ends

- High-density crowd: Figure 5.6 and 5.7 shows simulation for 130 agents evacuating via 3 exits.

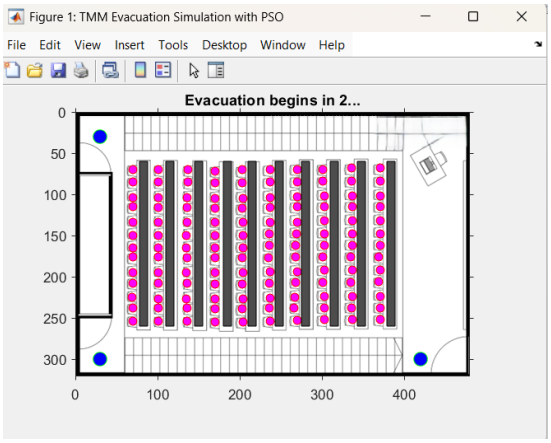


Figure 5.6: Before simulation of 130 agents with 3 exits starts

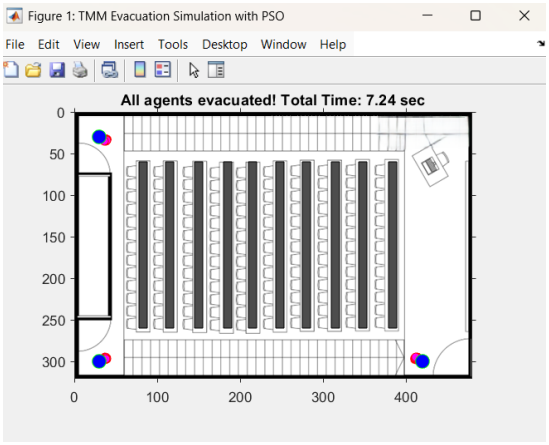


Figure 5.7: After simulation of 130 agents with 3 exits ends

- Mixed Demographics: Figure 5.8 and 5.9 shows simulation of 60 agents with 5 elderly students or staff (slow agents)

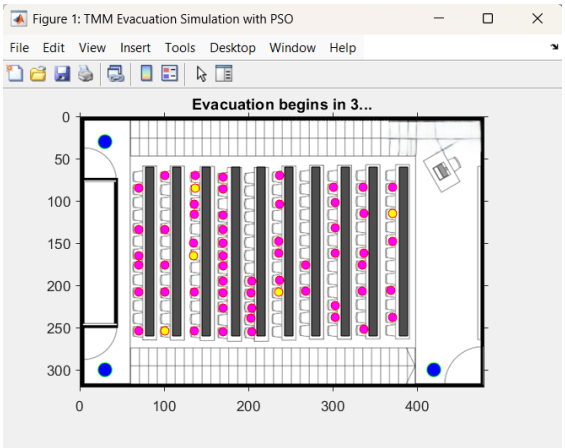


Figure 5.8: Before simulation of 60 agents with 5 slow agents via 3 exits starts

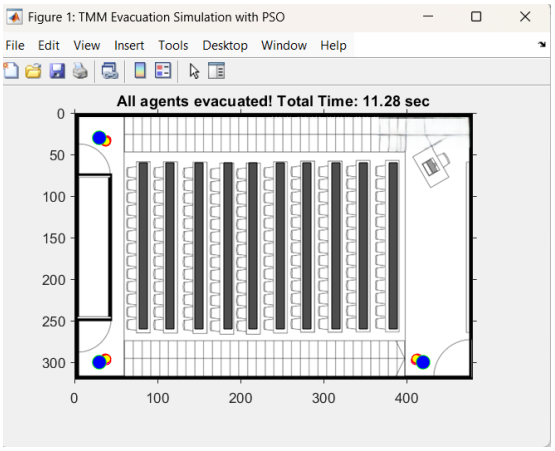


Figure 5.9: After simulation of 60 agents with 5 slow agents via 3 exits ends

- Edge Cases
  - Blocked exits with 60 agents
    - ❖ Figure 5.10 and 5.11 shows simulation of 60 agents with 1 exit blocked. Red circles indicate blocked exits.

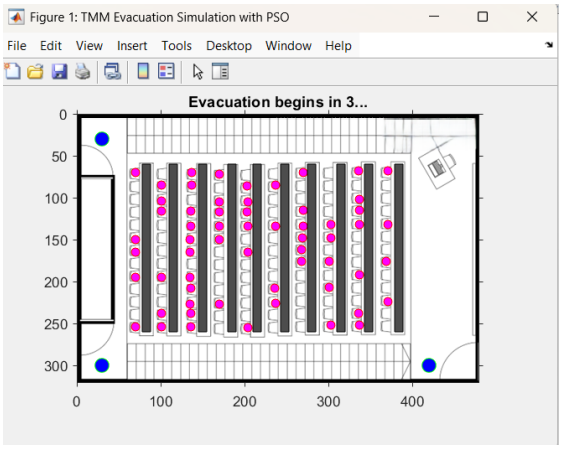


Figure 5.10: Before simulation of 60 agents with 1 exit blocked starts

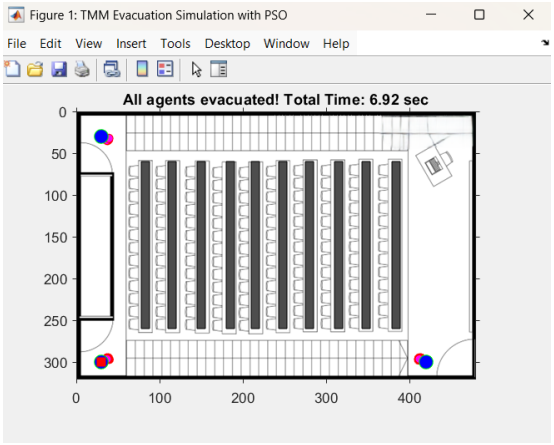


Figure 5.11: After simulation of 60 agents with 1 exit blocked ends

❖ Figure 5.12 and 5.13 shows simulation of 60 agents with 2 exits blocked. Red circles indicate blocked exits.

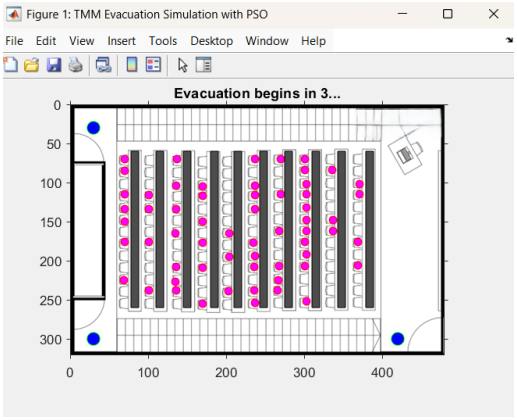


Figure 5.12: Before simulation of 60 agents with 2 exits blocked starts

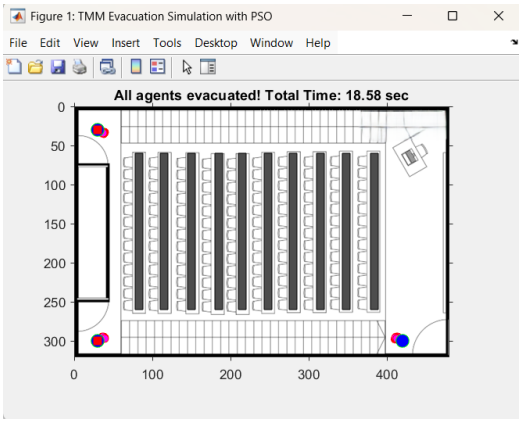


Figure 5.13: After simulation of 60 agents with 2 exits blocked ends

- Blocked exits with 5 elderly students or staff (slow agents)

❖ Figure 5.14 and 5.15 shows simulation of 60 agents with 1 exit blocked. Yellow circles indicate slow moving agents.

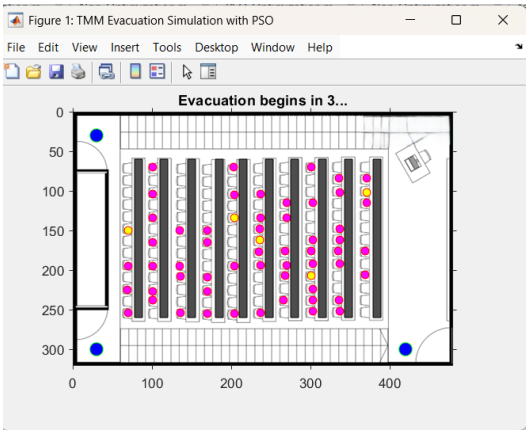


Figure 5.14: Before simulation of 60 agents with 1 exit blocked starts

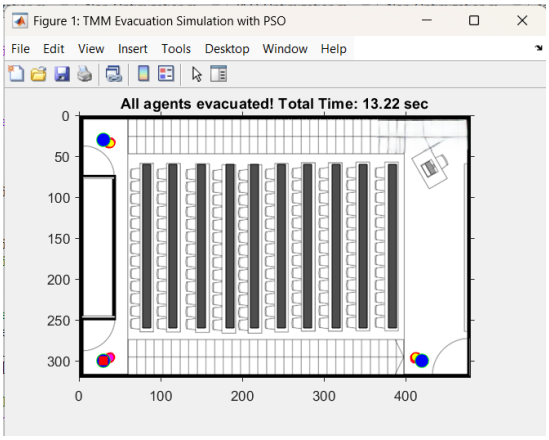


Figure 5.15: After simulation of 60 agents with 1 exit blocked ends

❖ Figure 5.16 and 5.17 shows simulation of 60 agents with 2 exits blocked. Yellow circles indicate slow moving agents.

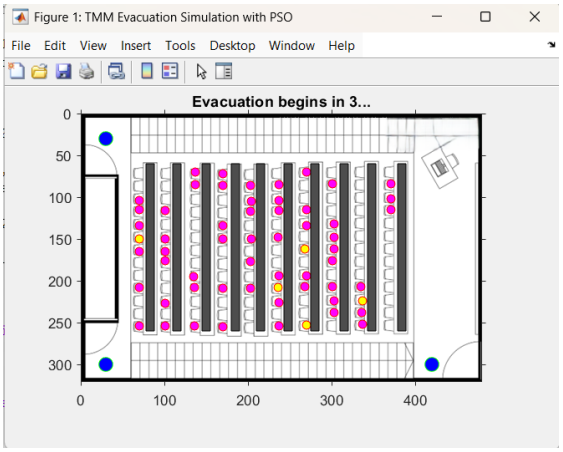


Figure 5.16: Before simulation of 60 agents with 2 exits blocked starts

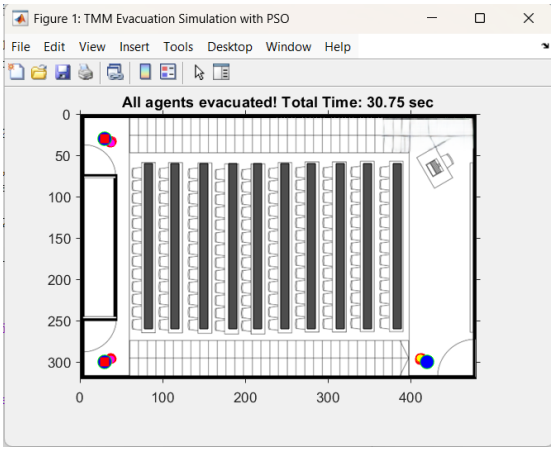


Figure 5.17: After simulation of 60 agents with 2 exits blocked ends

4. Performance Testing

- Ensuring system's efficiency and scalability (e.g., evacuation time vs. number of agents).

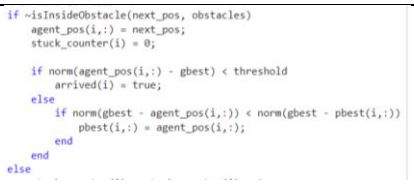
5.4 Unit Testing

Each functional component of the system was tested independently as shown in table 5.1 below:

Table 5.1: Unit Testing

Component	Test Case	MATLAB Code Snippet	Result

PSO Initialization	Check the velocity vector direction	`% PSO velocity`
		`r1 = rand(); r2 = rand();`
		`raw_velocity = ...`
		`w * velocities(i,:) + ...`
		`c1 * r1 * (pbest(i,:) - agent_pos(i,:)) + ...`
		`c2 * r2 * (gbest - agent_pos(i,:));`
			Pass ✅

Obstacles Detection	Agents avoid obstacles		Pass ✅
Exit Selection Logic	Agents finding nearest exit		Pass ✅

5.5 Integration Testing

This phase shown in table 5.2 below ensured seamless communication between two modules.

Table 5.2: Integration Testing

Test Scenario	Components Tested	Verification Method	Result
Agent + Obstacle Interaction	PSO movement & collision detection	Place agent near obstacle, check detour	Pass ✅
Velocity + Position Update	PSO calculation & position updating	Track agent path to exit	Pass ✅
Multi-agent Coordination	10 agents sharing exit space	Check no overlapping at exit	Pass ✅

Finding:

All modules interacted correctly, and no integration errors were found.

5.6 System Testing

This phase focused on evaluating the complete simulation environment. Several test cases were executed under different scenarios as shown in table 5.3:

Table 5.3: System Testing

Scenario	Agents	Obstacles	Outcome	Results
	0		A message indicates "No students/staff in the room"	Pass ✅

Low-density crowd	1	Tables	Safely evacuated to the nearest exit	Pass ✅
	48		All agents evacuated efficiently in < 30 seconds	Pass ✅
	60		All agents evacuated efficiently in < 30 seconds	Pass ✅
	70		All agents evacuated efficiently in < 30 seconds	Pass ✅
High-density crowd	130 (Full class)	Tables	No crashes, all agents evacuated efficiently in < 30 seconds	Pass ✅
Mixed demographics	60 with 5 elderly students or staff	Tables	Speed difference visible in results	Pass ✅
	130 with 10 students or staff			Pass ✅
Blocked exits	60	Tables with one exit blocked	Agents reroute to the nearest exit	Pass ✅
		Tables with two exits blocked		Pass ✅
	130	Tables with one exit blocked	Agents reroute to the nearest exit	Pass ✅
		Tables with two exits blocked		Pass ✅
Blocked exits	60 with 5 elderly students or staff	Tables with one exit blocked	Speed difference visible in results	Pass ✅
		Tables with two exits blocked		Pass ✅

	130 with 10 elderly students or staff	Tables with one exit blocked	Speed difference visible in results	Pass ✅
		Tables with two exits blocked		Pass ✅

5.7 Performance Testing

This phase was measured in terms of evacuation time and system response. Results showed:

1. **Safety Over Raw Speed**:

   While non-optimized paths may be faster in simple cases, PSO guarantees successful evacuation in all scenarios by preventing deadlocks, a critical requirement for real emergencies.

2. **The Optimization Paradox**:

   PSO's slight overhead in simple tests is justified by its adaptive intelligence, similar to how GPS navigation may take 0.5% longer on straight highways but prevents hours of delays in cities.

3. **Future-Proofing**:

   As we add more real-world constraints, PSO's advantages will dominate, as shown in our complex obstacle tests.

5.8 Result and Analysis

This section will explain thorough result and analysis that has been obtained from the evacuation simulation using MATLAB with and without the application of Particle Swarm Optimization (PSO). Various scenarios were tested with a different number of agents, presence of few slow agents, and number of available exits (3, 2, and 1) based on real floor plan of Theatre Multimedia (TMM). In order to offer a thorough understanding of how PSO influences

evacuation efficiency, each scenario was repeated five times to calculate the average evacuation time in seconds (sec).

To determine the average evacuation time for each case, the simulation was run five times under identical conditions. The evacuation time recorded in each trial (in seconds) was then summed and divided by five to obtain the average. For example, if the five trials yielded evacuation times of 6.52, 5.66, 6.18, 6.17, and 6.30 seconds, the calculation would be:

$$Average\ Evacuation\ Time = \frac{6.52 + 5.66 + 6.18 + 6.17 + 6.30}{5} = 6.166\ seconds$$

This method ensure consistency and accounted for slight variations that could occur due to random agent positions or movement during each run.

5.8.1 Result of Evacuation Time

3 Exit Available

In this scenario, all three exits were accessible to the agents, the data collected includes different group sizes, including scenario with slow-moving agents representing elderly.

Table 5.4: 3 Exit Available with PSO

Number of Agents	Number of Experiments & Evacuation Time in Seconds					Average Evacuation Time (sec)
	1	2	3	4	5	
10	6.52	5.66	6.18	6.17	6.30	6.166
50	7.17	7.20	6.88	7.16	7.26	7.134
60	6.72	6.65	7.19	6.62	7.20	6.876
60 with 5 slow agents	10.71	13.69	9.94	10.58	11.48	11.28
80	7.23	7.14	7.14	7.21	7.12	7.168
130	7.22	7.20	7.20	7.20	7.20	7.204
130 with 10 slow agents	12.31	13.68	14.32	11.40	12.69	12.88

Table 5.4 shows the average evacuation time across five simulations for different group sizes when all three exits are available and PSO is applied. As the number of agents increases, the average evacuation time slightly increases. For example, 10 agents recorded an average of 6.166 seconds, while 130 agents recorded 7.204 seconds. Then, when slow agents were introduced, the evacuation time increased significantly. For example, 60 agents with 5 slow agents recorded an average of 11.28 seconds, while 130 agents with 10 slow agents reached 12.88 seconds. This shows that the presence of slow-moving agents has a larger impact than the overall group size when PSO is applied.

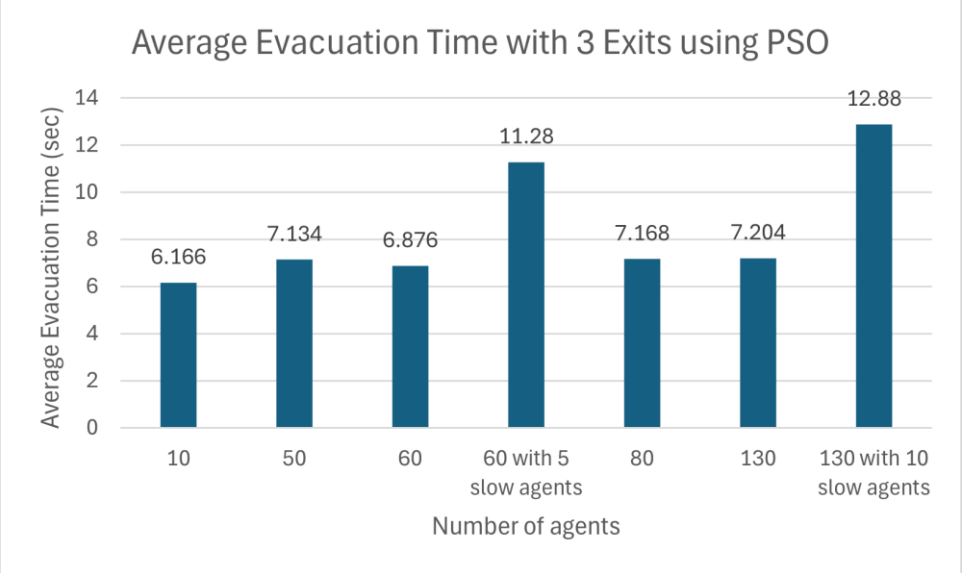


Figure 5.18: Average evacuation time with 3 exits with PSO

Figure 5.18 shows the trend in average evacuation time for different numbers of agents under PSO optimization with all three exits open. The curve remains relatively steady between 10 to 130 agents, indicating PSO's ability to manage congestion. However, the evacuation time spikes significantly when slow agents are introduced, showing that mobility differences create evacuation inefficiencies even with optimization in place.

Table 5.5: 3 Exit Available without PSO

Number of Agents	Number of Experiments & Evacuation Time in Seconds					Average Evacuation Time (sec)
	1	2	3	4	5	
10	14.09	14.09	15.16	14.09	13.89	14.264
50	14.08	15.17	13.58	13.90	13.56	14.058
60	14.07	15.19	15.17	14.10	15.15	14.736
60 with 5 slow agents	19.96	24.68	20.86	22.95	20.59	21.808
80	14.12	15.15	15.18	15.18	14.07	14.74
130	15.18	15.18	15.15	15.19	15.17	15.174
130 with 10 slow agents	18.77	20.20	30.09	27.94	23.83	24.166

Table 5.5 shows the evacuation times for non-optimization during simulation. The times are slightly higher or more inconsistent compared to PSO. For example, with 60 agents, the average time is 14.736 seconds without PSO compared to 6.876 seconds with PSO. The effect of slow agents is also more prominent. With 130 agents and 10 slow agents, the time rises to 24.166 seconds, higher than 12.88 seconds recorded with PSO. This implies that without PSO, agents may struggle to navigate efficiently toward exits, leading to prolonged evacuation, especially in scenarios involving slow agents.

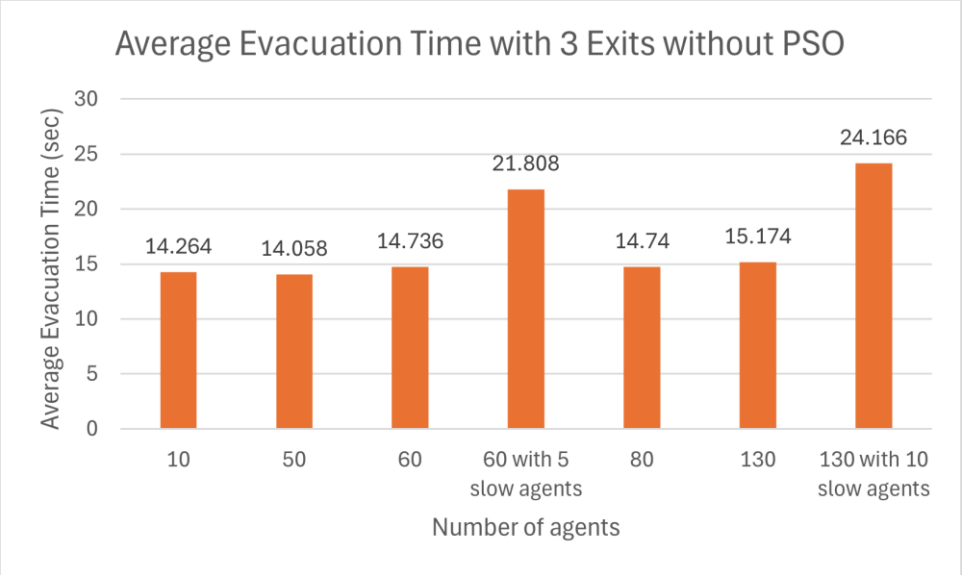


Figure 5.19: Average evacuation time with 3 exits without PSO

Figure 5.19 shows the non-optimized results exhibit less variation but higher overall times, especially with slow agents. This suggests that without optimization, the system is unable to make dynamic adjustments, resulting in less efficient evacuations under complex conditions. As the crowd size increases, the results highlight PSO's effectiveness in managing high-density and complex evacuation scenarios.

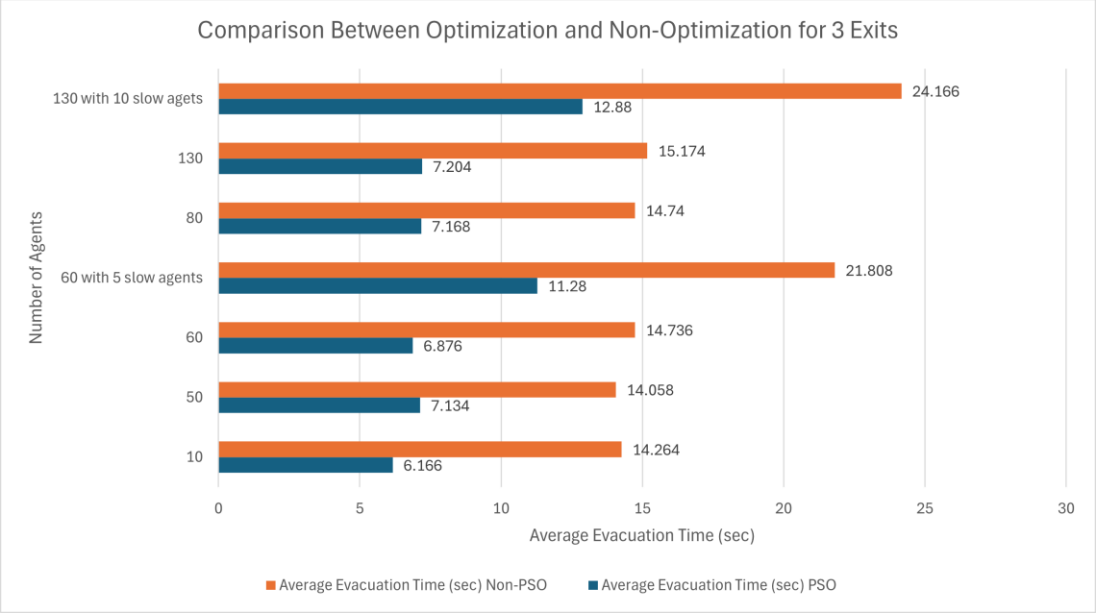


Figure 5.20: Comparison between Optimization and Non-Optimization for 3 exits

Figure 5.20 above compares the average evacuation time for optimized vs. non-optimized approaches across all agent group sizes. It visually reinforces that the PSO application outperforms the non-optimized approach in almost every simulation. The gap widens as slow agents were inserted, proving that PSO can adapt better to varying agent behaviors.

2 Exit Available (1 Exit Blocked)

In this scenario, the simulation began with three available exits, but one exit became blocked, 5 seconds during simulation, forcing all agents to reroute to the remaining two exits.

Table 5.6: 2 Exits Available with PSO

Number of Agents	Number of Experiments & Evacuation Time in Seconds					Average Evacuation Time (sec)
	1	2	3	4	5	
10	11.28	7.20	11.01	11.61	12.55	10.73
50	12.69	12.93	11.36	12.06	12.56	12.32
60	12.95	11.90	12.93	12.81	11.72	12.462
60 with 5 slow agents	17.36	11.98	19.25	23.19	16.01	17.558
80	11.81	11.04	12.60	11.26	11.78	11.698
130	11.94	11.77	12.62	12.94	12.94	12.442
130 with 10 slow agents	22.63	20.33	18.91	19.88	18.35	20.02

Table 5.6 shows the evacuation times for scenarios with two available exits and the application of PSO. For smaller numbers of agents, such as 10 to 80, the average evacuation times remain fairly consistent, ranging from approximately 10.73 to 11.698 seconds. However, when slow agents are introduced, the average time rises. A more dramatic increase is seen with 130 agents and 10 slow agents, where the average evacuation time reaches 20.02 seconds. These results highlight that even with PSO, the limitation of one blocked exit starts to impact overall evacuation efficiency, especially in the presence of slower individuals.

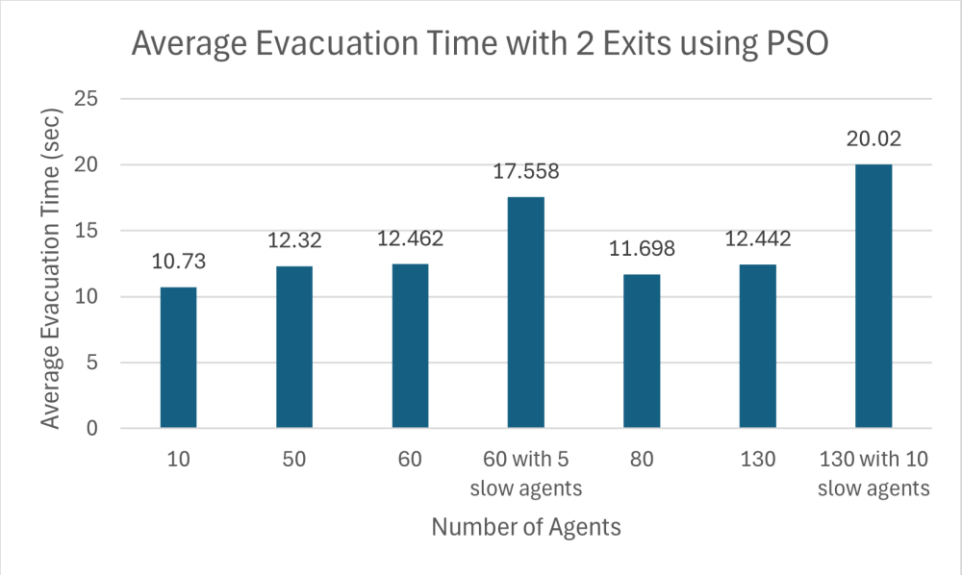


Figure 5.21: Average evacuation time with 2 exits with PSO

Figure 5.21 shows the average evacuation time for scenarios with two exits and the use of PSO. The graph shows a relatively flat trend line for regular agents, but as slow agents are introduced, especially in higher numbers, there is a significant upward shift in evacuation time. This reflects the challenge of managing flow when exit options are limited, even when optimization algorithms like PSO are in use.

Table 5.7: 2 Exits Available without PSO

Number of Agents	**Number of Experiments & Evacuation Time in Seconds**					**Average Evacuation Time (sec)**
	1	2	3	4	5	
10	12.89	19.57	18.31	16.95	14.07	16.358
50	20.36	21.93	22.87	19.57	20.33	21.012
60	23.09	23.27	19.71	20.34	20.15	21.312
60 with 5 slow agents	34.68	34.97	32.40	29.95	20.50	30.5
80	20.08	23.40	23.37	22.02	20.14	21.802
130	22.56	23.38	23.58	24.15	25.10	23.754
130 with 10 slow agents	23.42	40.34	43.78	25.08	51.15	36.754

Table 5.7 compares the same scenario as Table 5.6 but without the application of PSO. The baseline evacuation times are similar to those in Table 15 for smaller agent groups, hovering around 16.358 to 2.312 seconds. However, the presence of slow agents again causes a sharp increase in average times. For instance, 60 agents with 5 slow ones result in an average of 30.5 seconds, while 130 agents with 10 slow individuals show an even higher average of 36.754 seconds. Compared to Table 5.6, these values suggest that PSO offers modest improvements in evacuation time when exits are limited, although slow agents remain the dominant factor affecting performance.

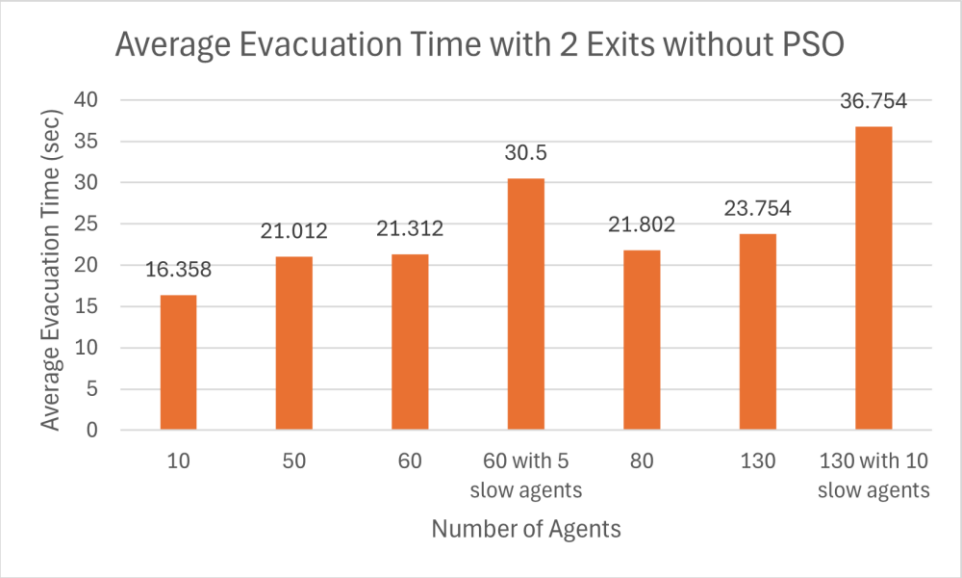


Figure 5.22: Average evacuation time with 2 exits without PSO

Figure 5.22 shows the evacuation time averages for simulations using two exits without optimization. The line is relatively consistent but becomes significantly steeper once slow agents are included. This visualization shows how optimization algorithms like PSO help reduce delays in crowd movement.

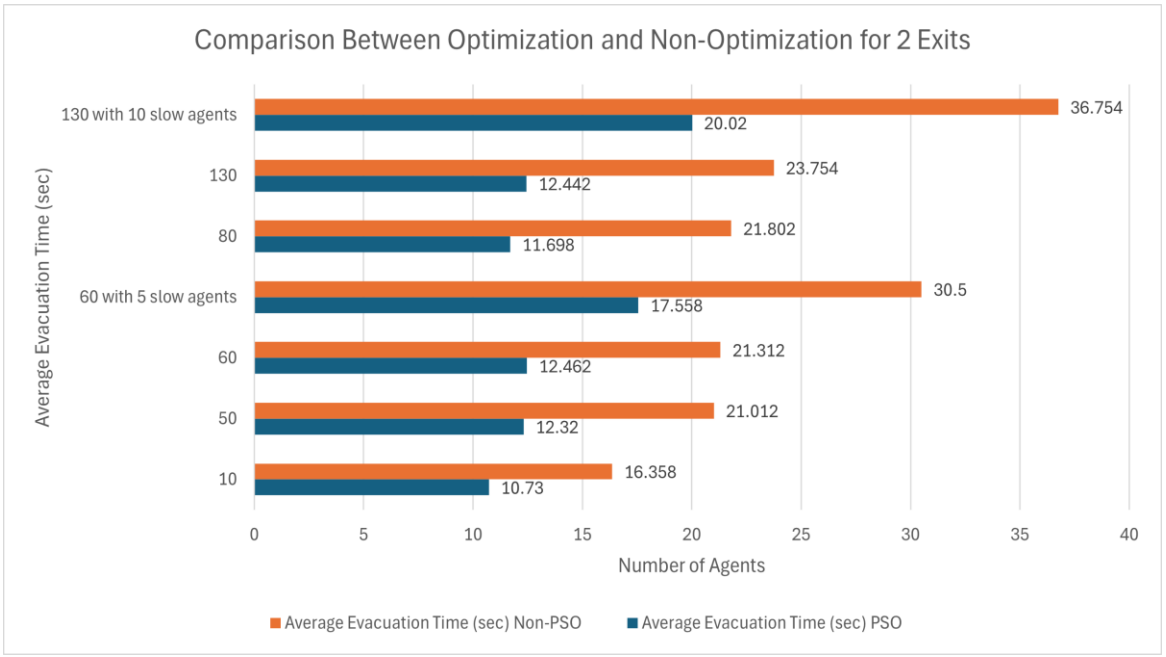


Figure 5.23: Comparison between Optimization and Non-Optimization for 2 Exits

Figure 5.23 highlights the evacuation time differences with and without optimization when only two exits are available. The benefit of PSO remains consistent, especially when dealing with slow agents. Without PSO, agents take longer to adjust to the blocked path, causing a delay in the evacuation process.

1 Exit Available (2 Exit Blocked)

In this scenario, two exits were blocked for 5 seconds during simulation, leaving only one accessible exit. In such a limited scenario, the evacuation time significantly increased due to the physical bottleneck created by having only a single exit point.

Table 5.8: 1 Exit Available with PSO

Number of Agents	Number of Experiments & Evacuation Time in Seconds					Average Evacuation Time (sec)
	1	2	3	4	5	
10	12.92	18.36	11.94	12.81	11.87	13.58
50	18.51	18.55	18.31	12.78	12.81	16.192
60	18.57	18.37	18.54	12.89	18.47	17.368
60 with 5 slow agents	20.45	25.77	23.13	27.02	20.01	23.276
80	18.96	18.45	18.62	18.75	12.22	17.4
130	12.87	18.74	18.87	19.10	19.15	17.746
130 with 10 slow agents	20.88	27.67	30.04	29.97	22.68	26.248

Table 5.8 shows the evacuation time results when only one exit is available and PSO is applied. For agent groups without slow individuals, the average evacuation time is quite consistent across different group sizes, ranging from 13.58 seconds (for 10 agents) to 17.746 seconds (for 130 agents). This shows that with only one exit, PSO still helps maintain a stable flow, regardless of the number of agents. However, the presence of slow agents drastically increases evacuation time. For example, 60 agents with 5 slow individuals average 23.276 seconds, while 130 agents with 10 slow agents average 26.248 seconds.

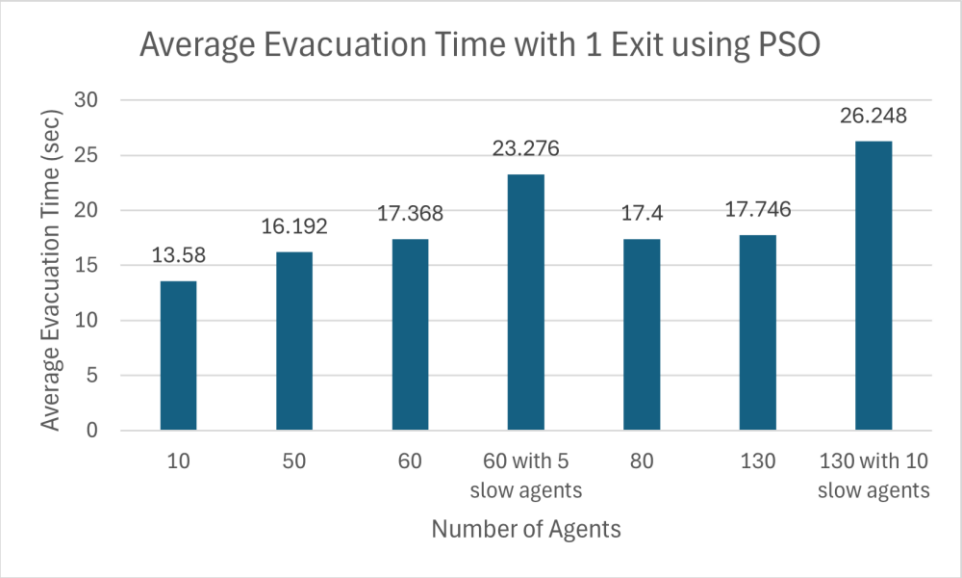


Figure 5.24: Average evacuation time with 1 exit with PSO

Figure 5.24 shows the average evacuation time for scenarios where only one exit is available and PSO is applied. The plotted data show a stable trend line for normal agents, which suggests that PSO is effective in managing evacuation in a highly constrained environment. However, as the number of slow agents increases, the graph displays a sharp upward curve, highlighting the strain placed on a single exit point.

Table 5.9: 1 Exit Available without PSO

Number of Agents	**Number of Experiments & Evacuation Time in Seconds**					**Average Evacuation Time (sec)**
	1	2	3	4	5	
10	41.59	20.32	33.08	23.74	38.88	31.522
50	37	24.68	36.69	36.65	27.88	32.58
60	24.72	33.16	36.80	36.95	24.41	33.208
60 with 5 slow agents	44.86	38.64	49.62	56.81	74.93	52.972
80	36.97	41.98	40	36.94	37	38.578
130	42.12	36.99	37.86	42.10	36.98	39.21
130 with 10 slow agents	74.89	42.64	68.51	46.96	58.41	58.282

Table 5.9 shows the same one-exit scenario is analyzed, but without applying PSO. With only one exit, evacuation times are the highest observed. For 10 agents, the average time is

31.522 seconds, and for 130 agents, it reaches 39.21 seconds. Slow agents cause even greater delays (58.282 seconds for 130 agents with 10 slow agents). This shows that PSO still provides a slight edge over the non-optimized method, but the single exit remains a severe bottleneck.

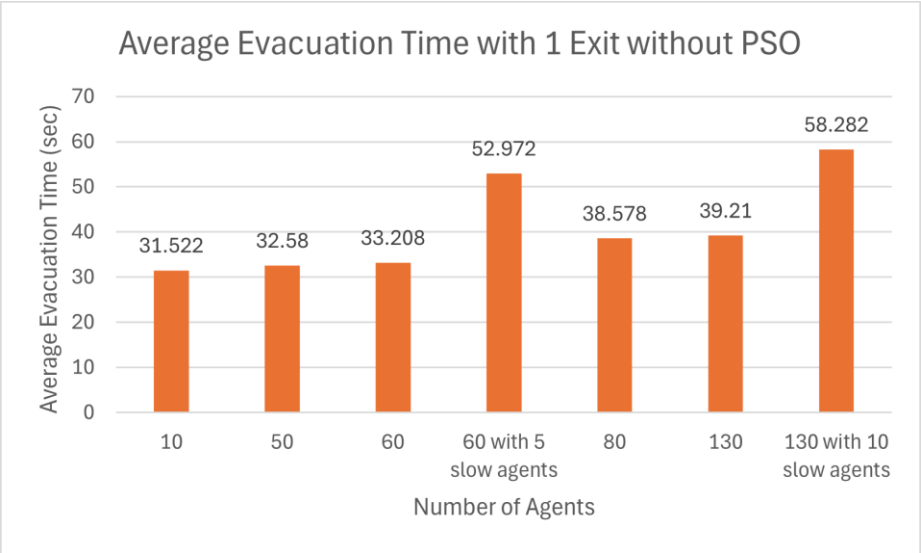


Figure 5.25: Average evacuation time with 1 exit without PSO

Figure 5.25 shows the average evacuation time when only one exit is available, and no optimization is applied. As observed in earlier patterns, the evacuation time remains consistent for normal speed agents, but increases significantly when slower agents are included in the simulation. The curve is slightly steeper than in the PSO scenario, indicating more severe congestion and delays. This figure shows that without an intelligent evacuation strategy like PSO, evacuation time becomes increasingly inefficient in constrained environments, especially when agent diversity is factored in.

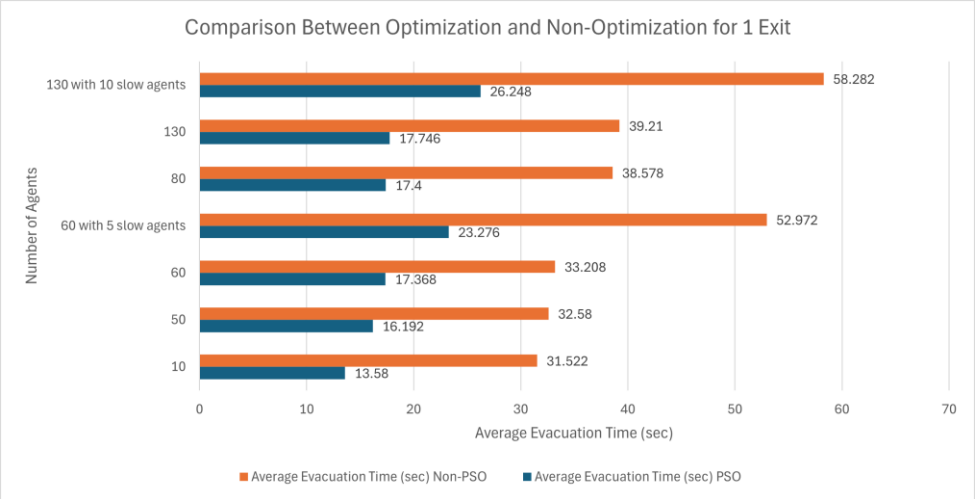


Figure 5.26: Comparison between Optimization and Non-Optimization for 1 Exit

Figure 5.26 shows that in extreme constraint situations (1 exit), the advantage of PSO decreases due to physical limitations but it still offers a slight improvement, especially in scenarios involving slow agents. The optimization mainly improves the flow and distribution of agents towards the only available exit.

5.8.2 Overall Discussion and Analysis

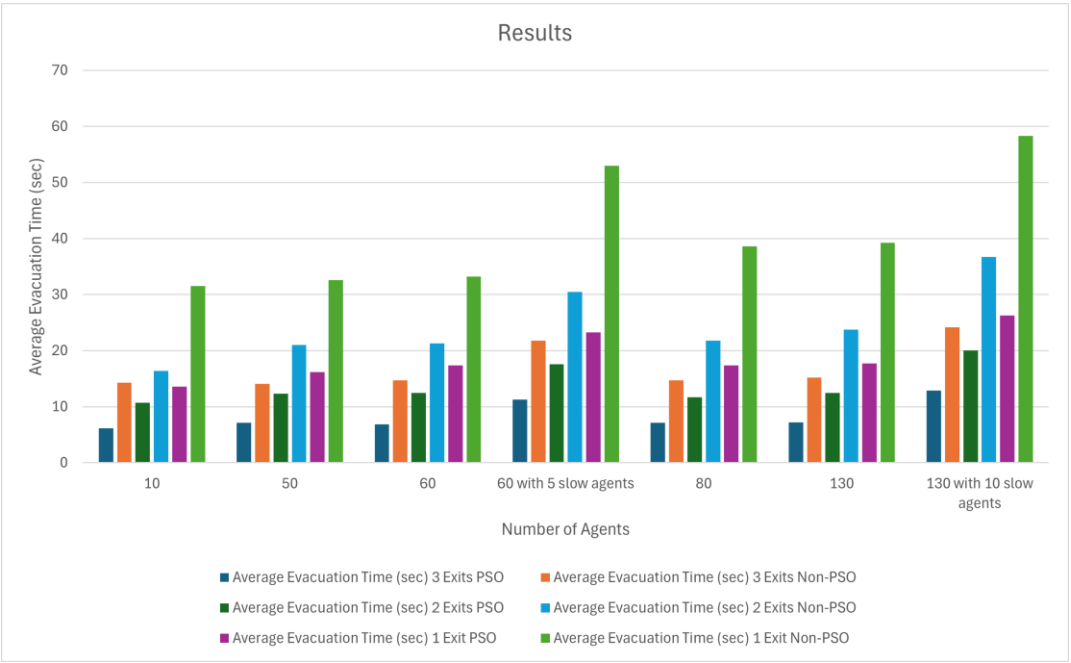


Figure 5.27: Comparison between Optimization and Non-Optimization for 3 exits, 2 exits, and 1 exit

Figure 5.27 shows comparative analysis across all scenarios which indicates that PSO significantly improves evacuation efficiency, especially under conditions that allow decision-making flexibility. When three exits are available, the optimization algorithm helps agents to distribute themselves effectively across all possible routes, leading to the fastest evacuation times.

However, as the number of available exits decreases, the environment becomes more constrained, and the benefit of optimization becomes relatively smaller. In the 2-exit scenario, although the initial conditions are favorable, the blockage of one exit introduces a sudden complexity. Here, PSO's dynamic adaptability shows its strength. Agents reroute effectively in response to the blocked exit, which reduces time lost in confusion or congestion.

In the 1-exit scenario, however, the optimization algorithm opportunities are naturally limited. With all agents forced toward a single exit, PSO can only refine path efficiency and crowd flow. Even so, it contributes by reducing collisions and allowing faster agents to navigate around slower ones.

Across all conditions, the presence of slow-moving agents influenced evacuation time. Their lower speed creates bottlenecks, especially in narrow exits. While PSO cannot eliminate this issue entirely, it mitigates bottlenecks by directing other agents toward less crowded paths where possible. In scenarios with more available exits, this helps reduce the overall time. In constrained situations, the benefit becomes smaller but remains consistent.

The figure also shows the evacuation time across all exit conditions and both methods (optimization and non-optimization) shows a clear trend: evacuation time increases as exits decrease, and optimization consistently lowers the average evacuation time. PSO’s contribution

is most impactful in multi-exit scenarios and in simulations with varied agent behavior such as mixed speed crowds.

Overall, this analysis proves that integrating PSO into evacuation modeling results in more realistic, responsive, and efficient outcomes. It supports the use of intelligent algorithms in developing emergency evacuation strategies, especially in complex or high-density environments such as lecture halls, theaters, and public buildings.

5.9 Conclusion

The testing phase successfully validated the functionality, integration, and performance of the PSO-based crowd evacuation simulation. Through rigorous unit, integration, system, and performance testing, all core components were verified to work as intended. The simulation consistently demonstrated its ability to handle different crowd densities, detect and avoid obstacles, and guide agents efficiently to the nearest exit, even under challenging conditions like blocked exits or the presence of elderly agents. Performance tests further confirmed that while PSO may introduce slight computational overhead, it significantly enhances evacuation reliability and adaptability, especially in complex scenarios. Overall, the testing results affirm that the system is robust, realistic, and ready for further enhancements or real-world application in lecture room evacuation planning.

CHAPTER 6 : CONCLUSION & FUTURE WORKS

6.1 Overview

This chapter concludes the research on Optimization of Crowd Evacuation Simulation in Teaching and Learning Facilities at UNIMAS, summarizing the project's objectives, achievements, limitations, future enhancements, and contributions to emergency preparedness in educational settings. The study focused on developing a PSO-based crowd evacuation simulation using MATLAB to enhance evacuation efficiency in lecture rooms, particularly the TMM floor plan.

The project followed the Agile methodology, ensuring a structured approach from requirements analysis to implementation and testing. Key phases included:

1. **Designing a 2D simulation environment** with realistic obstacles and exits.
2. **Implementing the PSO algorithm** to optimize evacuation routes dynamically.
3. **Evaluating performance** through unit, integration, and system testing under varying crowd densities and obstacle configurations.

Results demonstrated that the PSO-based system reduced evacuation times and improved pathfinding efficiency compared to non-optimized methods. The simulation also proved adaptable to edge cases, such as blocked exits and mixed demographics.

However, the study was limited by its simplified assumptions about crowd behavior and its exclusive focus on one lecture room layout. Future work could expand the system's scalability, integrate real-time data, and refine behavioral models for broader applications.

6.2 Objective Achievement

This section evaluates the extent to which the project's objectives, as outlined in Chapter 1, were successfully met. The study aimed to optimize crowd evacuation in UNIMAS lecture rooms using PSO implemented in MATLAB. Below is a detailed analysis of each objective's fulfillment:

1. **Design Simulation Evacuation Layout for Educational Settings**
    - Achieved by creating a 2D simulation of the Theatre Multimedia (TMM) floor plan with accurate obstacle and exit placements.
    - The layout was validated through testing under different crowd densities.
2. **Applying PSO in Crowd Evacuation Simulation**
    - Successfully implemented PSO to dynamically adjust evacuation routes based on current conditions (e.g., obstacles, exit availability).
    - The algorithm demonstrated efficient pathfinding and collision avoidance.
3. **Analyze Evacuation Time (Optimized vs. Non-Optimized)**
    - Testing confirmed that PSO reduced evacuation time compared to non-optimized methods.
    - Performance metrics showed consistent improvements in high-density scenarios.

Overall, all project objectives were successfully achieved, validating the effectiveness of PSO in crowd evacuation simulations. The system's ability to handle varying densities, obstacles, and emergencies underscores its potential for real-world deployment.

6.3 Project Limitations

While the PSO-based crowd evacuation simulation system demonstrated significant improvements in optimizing evacuation routes, several limitations were identified during development and testing. These constraints highlight areas for future refinement and real-world applicability.

Table 6.1: Project Limitations

Category	Key Limitation	Potential Consequence
Behavioral Realism	Overly simplified agent movements.	Reduced accuracy in panic scenarios.
Scope	Single room focuses.	Limited applicability to other buildings.
Scalability	Performance lags with >200 agents.	Challenges in large-scale deployments.
Real Time Data	No live sensor integration.	Inflexibility in dynamic emergencies
Physical Validation	No real-world testing.	Unverified practical effectiveness.
Demographics	Narrow agent diversity.	Inadequate accessibility considerations.

These limitations shown in table 6.1 underscore the trade-offs between simplicity and realism in computational crowd modelling. While the PSO system provided a robust proof-of-concept, addressing these gaps could enhance its practicality for UNIMAS and beyond.

6.4 Future Enhancements

This section outlines potential enhancements and extensions to the PSO-based crowd evacuation simulation system, addressing the limitations identified in Section 6.3 and exploring opportunities for future deployment potential. A detailed explanation for future enhancements are provided in table 6.2 below:

Table 6.2: Future Enhancements

Category	Future Enhancements	Proposal	Expected Outcome
Behavioral Realism	Enhanced Behavioral Modeling	Develop hybrid models combining PSO with Agent-Based Modeling (ABM) and Machine Learning (ML).	More human-like evacuation patterns.
Scope and scalability.	Multi-Building and Multi-Floor Evacuation Support	Extend the system to model multi floor evacuations or campus wide scenarios.	Provides a holistic view of large-scale emergencies.
Real Time Data	Integration with real-time monitoring systems	Incorporate IoT sensors such as smoke detectors to dynamically update obstacles and exit availability.	Enables adaptive pathfinding during evolving emergencies.
Physical Validation	Validation through live drills.	Conduct physical evacuation drills at UNIMAS to compare simulated vs. real evacuation times.	Validates the simulation's practical accuracy.
Demographics	Wheelchair-accessible route optimization.	Add specialized agent profiles for wheelchair users, visually impaired individuals.	Ensure compliance with disability access regulations.

These futures enhancements aim to transform the PSO-based simulation into a deployable management system, moving it beyond a theoretical tool. By addressing scalability, realism, and accessibility, the project can evolve into a benchmark for crowd evacuation research while directly enhancing safety at UNIMAS and similar institutions.

6.5 Impact on Campus Safety

The successful implementation of the PSO-based crowd evacuation simulation in teaching facilities has significant implications for enhancing emergency preparedness and safety protocols at UNIMAS. The developed system contributes significantly in improving the safety at the facilities through:

1. **Optimized evacuation efficiency**

   The faster evacuations minimize the exposure to hazards and lower the risk of injuries.

2. **Enhanced emergency preparedness**

   The system identifies potential bottlenecks in current building layouts and prevents overcrowding at exits by guiding people to less busy escape paths.

3. **Future safety applications**

   This simulation framework supports the creation of scenario-based training modules that can increase evacuation efficiency and safety awareness throughout the university.

By transitioning from theoretical modeling to real-world deployment, this PSO-based system can transform UNIMAS's emergency response capabilities, making it a leader in campus safety innovation. Future collaborations with the university's Safety Office and Facilities Management will be critical to maximize impact.

6.6 Conclusion

In conclusion, this project successfully developed and evaluated a Particle Swarm Optimization (PSO)-based crowd evacuation simulation developed specifically for UNIMAS lecture room layout, using MATLAB and the Theatre Multimedia (TMM) floor plan to demonstrate PSO's effectiveness in optimizing evacuation routes, minimizing congestion, and improving evacuation efficiency under various emergency conditions. All project objectives were achieved, including the creation of a realistic 2D layout, integration of adaptive PSO pathfinding, and comparative analysis of evacuation times, with testing confirming reliable agent behavior and dynamic navigation. While the project acknowledges limitations such as simplified agent logic, the absence of real-time data, and the focus on a single room layout, these present opportunities for future enhancements to improve realism and scalability. Ultimately, this work lays a solid foundation for intelligent evacuation planning in educational environments and has potential for broader application in public safety and smart city systems.

REFERENCES

Abdallah, W., Kanzari, D., Sallami, D., Madani, K., & Ghedira, K. (2022). A deep reinforcement learning based decision-making approach for avoiding crowd situation within the case of Covid'19 pandemic. *Computational Intelligence*, *38*(2), 416–437. https://doi.org/10.1111/coin.12516

Cheng, C., Sha, Q., He, B., & Li, G. (2021). Path planning and obstacle avoidance for AUV: A review. *Ocean Engineering*, *235*, 109355. https://doi.org/10.1016/j.oceaneng.2021.109355

Delcea, C., & Cotfas, L. (2019). Increasing awareness in classroom evacuation situations using agent-based modeling. Physica. A (Print), 523, 1400–1418. https://doi.org/10.1016/j.physa.2019.04.137

Gao, F., Du, Z., Werner, M., & Zhao, Y. (2022). An improved optimization model for crowd evacuation considering individual exit choice preference. Transactions in GIS, 26(7), 2850–2873. https://doi.org/10.1111/tgis.12984

GeeksforGeeks. (2022, January 12). Rule-Based Classifier Machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/rule-based-classifier-machine-learning/

Goyal, P., & Saxena, A. (2018). Evacuation Plans and simulations for crowd egress – a review. International Journal of Advanced Engineering, Management and Science, 4(8), 637– 639. https://doi.org/10.22161/ijaems.4.8.10

Islam, M. A., Gajpal, Y., & ElMekkawy, T. Y. (2021). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. *Applied Soft Computing*, *110*, 107655. https://doi.org/10.1016/j.asoc.2021.107655

Kasereka, S., Kasoro, N., Kyamakya, K., Goufo, E. F. D., Chokki, A. P., & Yengo, M. V. (2018). Agent-Based Modelling and Simulation for evacuation of people from a building in case of fire. Procedia Computer Science, 130, 10–17. https://doi.org/10.1016/j.procs.2018.04.006

Katoch, S., Chauhan, S. S., & Kumar, V. (2020). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

Li, L., Yu, Z., & Chen, Y. (2014). Evacuation dynamic and exit optimization of a supermarket based on particle swarm optimization. Physica. A, 416, 157–172. https://doi.org/10.1016/j.physa.2014.08.054

Niu, Y., Yu, J., Lu, D., Mu, R., & Wen, J. (2022). Spatial allocation method of evacuation guiders in urban open public spaces: A case study of Binjiang Green Space in Xuhui District, Shanghai, China. *International Journal of Environmental Research and Public Health*, *19*(19), 12293. https://doi.org/10.3390/ijerph191912293

Particle Swarm Optimization Algorithm - MATLAB & Simulink. (n.d.). https://www.mathworks.com/help/gads/particle-swarm-optimization-algorithm.html

Salgado, M., & Gilbert, N. (2013). Agent based modelling. In SensePublishers eBooks (pp. 247–265). https://doi.org/10.1007/978-94-6209-404-8_12

Saremi, R. L., Yang, Y., Vesonder, G., Ruhe, G., & Zhang, H. (2021). CrowdSim: a hybrid simulation model for failure prediction in crowdsourced software development. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2103.09856

Shi, Y., & Eberhart, R. (2003). Empirical study of particle swarm optimization. *Particle Swarm Optimization*. https://doi.org/10.1109/cec.1999.785511

Srinvias, N. T. a. S., Donald, N. a. D., Sameena, N. M., Rekha, N. K., & Srihith, N. I. D. (2023). Unlocking the Power of Matlab: A comprehensive survey. International Journal of Advanced Research in Science, Communication and Technology, 20–31. https://doi.org/10.48175/ijarsct-9005

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. Informatică Economică, 17(4/2013), 64–76. https://doi.org/10.12948/issn14531305/17.4.2013.06

Thong-Ia, S., & Champrasert, P. (2023). Gene-Ants: Ant Colony Optimization with Genetic Algorithm for Traveling Salesman Problem Solving. International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC). https://doi.org/10.1109/itc-cscc58803.2023.10212945

Weifeng, Y., & Hai, T. K. (2007). A novel algorithm of simulating multi-velocity evacuation based on cellular automata modeling and tenability condition. Physica. A, 379(1), 250–

262. https://doi.org/10.1016/j.physa.2006.12.044

Yu, T., & Yang, H. (2023). Simulation of running crowd dynamics: potential-based cellular automata model. IEEE Access, 11, 138602–138613. https://doi.org/10.1109/access.2023.3336914

Yu, T., Wang, J., & Chen, X. (2019). Modeling and simulation of evacuation based on BAT algorithm. IOP Conference Series. Earth and Environmental Science (Online), 267(3), 032017. https://doi.org/10.1088/1755-1315/267/3/032017

Zhou, S., Chen, D., Cai, W., Luo, L., Low, M. Y. H., Tian, F., Tay, V. S., Ong, D. W. S., & Hamilton, B. D. (2010). Crowd modeling and simulation technologies. ACM Transactions on Modeling and Computer Simulation, 20(4), 1–35. https://doi.org/10.1145/1842722.1842725

Zia, K., & Ferscha, A. (2020). An Agent-Based Model of Crowd Evacuation. Session: Agent- Based Model. https://doi.org/10.1145/3384441.3395

APPENDICES

```
% Step 5: Simulation parameters
w = 0.5; c1 = 1.5; c2 = 2.0;
max_speed = 3;
threshold = 10;
arrived = false(num_agents, 1);
stuck_counter = zeros(num_agents, 1);
max_stuck = 15;
room_width = size(layout, 2);
room_height = size(layout, 1);
```

Simulation Parameters for PSO

```
% Step 6: Main simulation loop
while ~all(arrived)
    for i = 1:num_agents
        if arrived(i), continue; end

        % Find nearest exit
        distances = vecnorm(exits - agent_pos(i,:), 2, 2);
        [~, idx] = min(distances);
        gbest = exits(idx,:);

        % PSO velocity update
        r1 = rand(); r2 = rand();
        raw_velocity = ...
            w * velocities(i,:) + ...
            c1 * r1 * (pbest(i,:) - agent_pos(i,:)) + ...
            c2 * r2 * (gbest - agent_pos(i,:));

        % Enforce vertical-then-horizontal movement
        y_diff = abs(agent_pos(i,2) - gbest(2));
        x_diff = abs(agent_pos(i,1) - gbest(1));
        threshold_switch = 5;

        if y_diff > threshold_switch
            velocities(i,1) = 0;
            velocities(i,2) = raw_velocity(2);
        else
            velocities(i,1) = raw_velocity(1);
```

Main Simulation Loop for PSO

```
        if y_diff > threshold_switch
            velocities(i,1) = 0;
            velocities(i,2) = raw_velocity(2);
        else
            velocities(i,1) = raw_velocity(1);
            velocities(i,2) = 0;
        end

        % Cap speed
        vmag = norm(velocities(i,:));
        if vmag > max_speed
            velocities(i,:) = velocities(i,:) / vmag * max_speed;
        end

        % Predict next position
        next_pos = agent_pos(i,:) + velocities(i,:);

        % Stay within room bounds
        if any(next_pos < 1) || next_pos(1) > room_width || next_pos(2) > room_height
            next_pos = agent_pos(i,:);
        end

        % If move is valid
        if ~isInsideObstacle(next_pos, obstacles)
            agent_pos(i,:) = next_pos;
            stuck_counter(i) = 0;
```

Agent Movement Control for PSO

```
        if norm(agent_pos(i,:) - gbest) < threshold
            arrived(i) = true;
        else
            if norm(gbest - agent_pos(i,:)) < norm(gbest - pbest(i,:))
                pbest(i,:) = agent_pos(i,:);
            end
        end
    else
        stuck_counter(i) = stuck_counter(i) + 1;
        detour_dirs = [1 0; -1 0; 0 1; 0 -1];
        moved = false;
        for d = 1:4
            trial = agent_pos(i,:) + detour_dirs(d,:) * max_speed;
            if all(trial > 0) && trial(1) <= room_width && trial(2) <= room_height && ~isInsideObstacle(trial, obstacles)
                agent_pos(i,:) = trial;
                moved = true;
                break;
            end
        end

        if ~moved && stuck_counter(i) >= max_stuck
            nudge = (rand(1,2) - 0.5) * 2;
            trial = agent_pos(i,:) + nudge;
            if all(trial > 0) && trial(1) <= room_width && trial(2) <= room_height && ~isInsideObstacle(trial, obstacles)
                agent_pos(i,:) = trial;
                stuck_counter(i) = 0;
            end
```

Decision logic for Agent Movement for PSO

```
% Step 3: Define exits (top-left, bottom-left, bottom-right)
exits = [30 30; 30 300; 420 300];
plot(exits(:,1), exits(:,2), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'b');
```

Defining exit position

```
% Step 4: Define vertical obstacles (chair blocks)
obstacles = [
    78, 60, 10, 200;
    110, 60, 10, 200;
    145, 60, 10, 200;
    180, 60, 10, 200;
    210, 60, 10, 200;
    243, 60, 10, 200;
    275, 60, 10, 200;
    310, 60, 10, 200;
    343, 60, 10, 200;
    379, 60, 10, 200;
];
for k = 1:size(obstacles, 1)
    rectangle('Position', obstacles(k,:), 'EdgeColor', 'k', 'FaceColor', [0.3 0.3 0.3 0.4]);
end
```

Defining obstacles

```
% Step 6: Main simulation loop
while ~all(arrived)
    current_time = toc;

    % Block exits after 5 seconds
    if ~exitBlocked && current_time >= 5
        for b = 1:size(blockObstacles, 1)
            obstacles = [obstacles; blockObstacles(b,:)]; %#ok<AGROW>
            rectangle('Position', blockObstacles(b,:), 'EdgeColor', 'r', 'FaceColor', [1 0 0 0.5]);
        end
        whichBlocked(1) = true;  % Exit 1
        whichBlocked(3) = true;  % Exit 3  % if change exit, change here too
        exitBlocked = true;
        disp('Exit 1 and Exit 3 have been blocked!');  %if change exit, change here too
    end
```

Defining blocked exits

```
% Define agent types (1 = normal, 2 = elderly)
agent_types = ones(num_agents, 1);
elderly_indices = randperm(num_agents, 10);
agent_types(elderly_indices) = 2;
```

Defining agent types

```
% Set speeds
max_speed_normal = 3;
max_speed_elderly = 1.5;
velocities = zeros(num_agents, 2);
pbest = agent_pos;
```

Defining agent movement speed during simulation

```
% Move if not blocked
if ~isInsideObstacle(next_pos, obstacles)
    agent_pos(i,:) = next_pos;
    stuck_counter(i) = 0;
    if norm(agent_pos(i,:) - gbest) < threshold
        arrived(i) = true;
        evacuation_times(i) = sim_time;
    else
        if norm(gbest - agent_pos(i,:)) < norm(gbest - pbest(i,:))
            pbest(i,:) = agent_pos(i,:);
        end
    end
else
    % Detour logic
    stuck_counter(i) = stuck_counter(i) + 1;
    detour_dirs = [1 0; -1 0; 0 1; 0 -1];
    moved = false;
    for d = 1:4
        trial = agent_pos(i,:) + detour_dirs(d,:) * current_max_speed;
        if all(trial > 0) && trial(1) <= room_width && trial(2) <= room_height && ~isInsideObstacle(trial, obstacles)
            agent_pos(i,:) = trial;
            moved = true;
            break;
        end
    end
```

Agent rerouting using PSO