Comparative Analysis between Micro and Macro Agent Evacuation Using PedSim Simulator

Hrishikesh Narayanankutty

Matriculation Number : 255670

A Dissertation

Presented to the Faculty

of University of L'Aquila

in Candidacy for the Degree

of Master of Science

Adviser: Professor Henry Muccini

Co-Supervisor: Mahyar Moghaddam

March 2019

Abstract

Disaster management pertains to coping with disaster, relief and evacuation in the event of a catastrophe. It is of paramount importance as it pertains to the protection of lives and property during the time of calamity. To mitigate the damage of such incidents, simulations are performed in order to formulate an efficient and optimized exit strategy for the people who are struck by such unfortunate incidents and to evacuate them to a safe zone as quickly as possible. The simulation of an evacuation of the participating agents broadly falls under two categories - Macro Agent Simulation and Micro Agent Simulation. Grouping of various agents may also be present in Micro Agent Simulation (according to age, gender, social ties such as family, friendships etc.). Extensive work has been carried out to simulate disaster scenarios comprising of a massive number of agents with an aggregate set of characteristics (Macro Agent Simulation); my thesis pertains to simulate social behaviour evacuation patterns of agents during a disaster scenario, especially pertaining to micro agent simulations. Realistic human and social behaviour characteristics are possible only through micro agent-based simulations. This is because complex psychological and sociological paradigms can be mapped to the agents under simulation and hence dynamic real-time strategy decisions can be better understood, especially during a crisis. Through this thesis, I aim to present the simulation and comparison of various micro and macro agent scenarios.

In order to run the aforementioned simulations, an open source microscopic pedestrian simulator - PedSim is used. This tool not only simulates the various complex scenarios. it also provides visual feedback in real time. PedSim is a crowd simulation library capable of analysis of real time pedestrian flow rate. This agent-based modeling (ABM) tool is a class of computational models for simulating the actions and interactions of autonomous agents (either individual or collective entities such as organizations or groups) with a view to assessing their effects on the system as a whole.

It combines elements of game theory, complex systems, emergency, computational sociology, multi-agent systems, and evolutionary programming. Although there are many proposed agent modelling simulators that are available, many if not most are with commercial license and do not support real time data flow. This tool is also customized and further extended in order to take an optimized routing algorithm for agents. The PedSim simulator is modelled for both indoor and outdoor areas (parking lots, forests etc.). The simulation tool is able to take sensory based data and apply them to the modelling agents/nodes to simulate real/design time analysis. The main advantage of this library is the architecture that enables visibility of users live using tcp/stream-based output through batch processing. The implementation is pure in C++ with minimal external dependencies like Qt Framework. The output of PedSim can be translated using a graphics engine to provide visually appealing realistic render of a walking person. The code is modular, scalable and open source available under GPL license.

The main goal of this thesis is to exhibit an objective performance analysis and a comparison between an existing macro simulation algorithm against an optimized simulation coupled with realistic constraints, testing them against a massively populated macro and micro agent model for design-time dynamic evacuation.

Acknowledgements

To my parents and my dear brother,

I am grateful for your undying love and support through the challenges I face. I am the person I am today because of you. I shall always cherish your wisdom and presence in my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the better part of the modern era, the increasing topological changes in urban environment have caused the human civilization to be subjected to an increasing risk of disasters due to natural or artificial, and/or a combination of the two causes. Hence it has become an ever increasing need to design infrastructure to handle such disasters in the event it may occur. But even more so, the safe evacuation of people and personnel with the premises of the affected infrastructure takes precedence when dealing the necessary mitigation and disaster risk management. The evacuation time of people from a scene of disaster is extremely crucial in the case of an emergency in disaster situation. In order to reduce the time taken for evacuation, better and more robust exit strategy evacuation algorithms are developed which are used to model participating agents for their exit patterns and exit strategies and clock them based on performance, efficiency and practicality. In order to evaluate such parameters, evacuation simulation models have been developed and constantly tested over the years to investigate the emergency egress capabilities of the built environment for a variety of reasons including: difficulty in conducting real evacuation tests (drills); aiding building design and confirming conformity to building regulations; and even determining optimal evacuation routes for the building occupiers [6]. Computational

models were first introduced in the early 1980s since the inception of computers. During the 90s, academic researchers aimed to improve the capabilities of these models in order to optimize and improve the pathfinding performance of the evacuees and their corresponding movements [19]. Over the course of many years, these algorithms have become the industry standard to perform exit evacuation time and performance analysis. The purpose of this thesis is to study the state of the art algorithm and to draw comparisons between micro model agent simulation with respect to its counterpart, the macro model agent simulation.

1.1 Motivation

Realistic simulations involve complex relationships between an individual and the surroundings. Three types of interactions are possible whilst the individual performs complex decision making during an evacuation scenario. Through this thesis I aim to optimize these three types of interactions through various constraints and real time elements. The following encounters may be classified as below [6]:

- People-people interactions - interactions between participating evacuees.

- People-structure interactions- interactions within the enclosed topology.

- People- environment interactions- interactions with quarantined atmosphere (fire, smoke etc.) and possible debris.

During disaster scenarios standard evacuation pathways are often rigid and cannot autonomously provide modification for an exit strategy as the disaster ensues. The evacuees often find themselves in situations that force them to rely on general guidelines about how to react in emergency evacuation [5]. While such dynamic hazards cannot all be dealt with, using the traditional approach, Lujak, M., Billhardt H., Dunkel, J. et al [23] attempts to help the evacuees adapt to the changing topography

of the environment due to hazard dynamics by updating a real time monitoring system using an IoT architecture built in place within the premises of the said topology of the building. The above mentioned work uses a combination of IoT devices to sense and identify and provide real-time monitoring for the participating agents. Our work also obtains data from real time sources and then it is simulated with both macro and micro based agent models and then used to analyse and draw conclusions from the resulting data.

1.2   Agent Based Model Simulation

The present thesis is a design of a computational software using Agent Based Model (ABM) to help speedy evacuation in emergency situations. Our software architecture help optimise the navigational flow rate in cases of real-time disaster management. The scenario is that of many victims are struck up in a very large building in the event of an occurrence of a disaster like fire, earthquake, poisonous chemical gas leakage, imminent bomb attack, potential imminent building collapse (similar to that of twin towers of 9/11). As a side benefit the experience acquired gives valuable information in the case of architecture design at the time of building construction itself. By conveying a suggestive path for each individual in the building at the time of disaster, the panic, erratic and groping movements are reduced to minimum helping to achieve a streamlined flow. Bottlenecks in the flow are avoided by redirecting people to alternate paths. The benefit of the overall perspective of the scenario and informed management is instantaneously conveyed to each and every individual in real time. The floor capacities and width of the doors and passageways form part of the constraints in the modelling. This is scalable and generic version.

1.3 PedSim Simulator

PedSim is a pedestrian simulator tool designed as a front end to facilitate disaster management scenarios. Disaster management plans are multi-layered and topology specific. Tools catering to disaster management require the users to input topographic information and also provide crowd/agent location information. This aggregate of data is then processed initially to create a layout of the premises. The layout defines several rules and boundary conditions imposing restrictions for the crowd to navigate. Such information can be used to instantly model very specific scenarios such as evacuation of specific floors of a building. Since, PedSim enables input of design specific topology, protocols can be established quickly for speedy evacuation. The second part includes behaviour analysis of crowd during different disaster patterns. The social behaviour of victims in disaster afflicted scenarios have been of keen interest to researchers over the recent years. Disaster affected regions often portray victims who exhibit severe trauma or stress leading to emotional/psychological shut down or present themselves making erratic decisions etc. This tool enables a platform for crowd flow modelling. Erratic movement patterns can be modelled, simulated and visualized to study the impact and speed of evacuation. Such processed information can be used to provide instructions or navigation guidelines for optimal evacuation. PedSim also allows for real time monitoring of crowds providing instant feedback on the visual front end. This enables redirection of crowds for balancing congestion while developing exit strategies. Each exit can also be associated with restrictive exit capacity and corresponding flow rate ensuring crowd balancing. The architectural design of the tool itself takes into account various factors such as streaming and batch processing of time critical data within permissible thresholds.

1.4 Objectives and Approach

My work pertains to application of PedSim on our university building to implement and analyze the impacts of crowd evacuation during a disaster. The university building taken into consideration for our present study is level 3 of building (Coppito 0). The area under consideration is divided into 18 blocks. Each block consists of a set number of cells. Each cell represents a cubicle or a room. There are 26 cubicles/rooms in all. There is also one conference hall and two generic multipurpose areas used for allied purposes. In addition, there are several hallways and pathways for navigation. This area has four key exits distributed across the different boundary walls of the premises. At any given time, each cell hosts atmost twenty agents, reaching a maximum during peak hours of 10am to 2 pm Italy time. The premises often on average hosts about 100 people, although sometimes more agents are assumed to be present. We architected pedestrian flow simulations during a generic disaster which can include earthquake, fire and chemical leakage. The data is used to generate crowd clusters during events other than disasters such as workshops, cafeteria gatherings, conference gatherings etc. The figure 1.1 shows a typical population cluster during a conference gathering.

This thesis presents a case study where the Coppito 0 building is simulated during a disaster scenario using grid geometry based navigation algorithm as presented in the An IoT Software Architecture for an Evacuable Building Architecture by H.Muccini et.al [24].
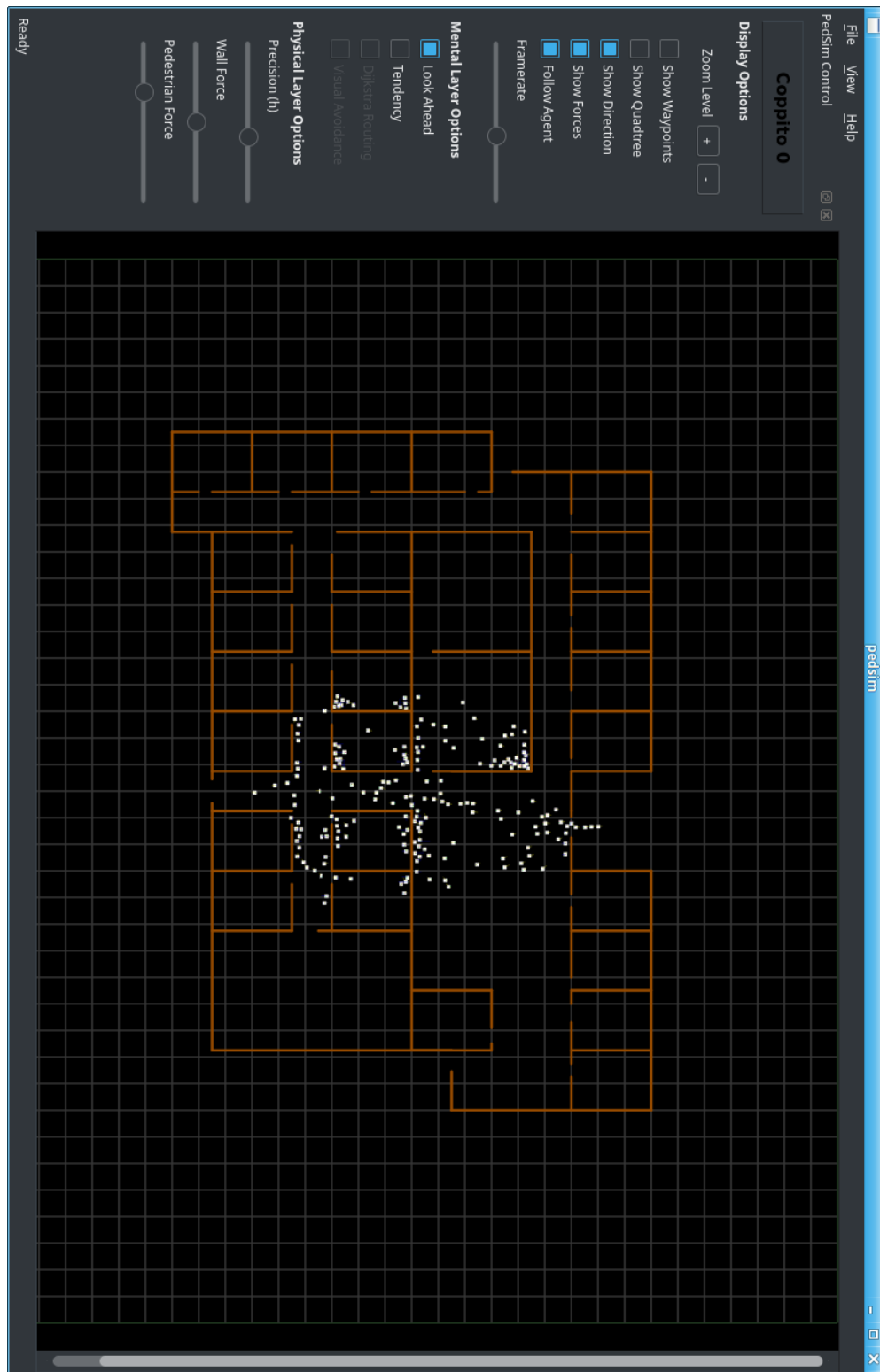
Figure 1.1: Agent Cluster Formation in Coppito 0 Building

The work also involves in expanding the capabilities of the existing PedSim simulation tool to include the new topology and the also to incorporate the algorithm and also further extended developed from the aforementioned paper and perform a real time analysis between micro and macro agents with the following criteria:

- specifying the cells by social distances

- respecting the area capacity and doors capacities constraints

- varying movement speed according to various groups

- simulating social attachment among some agents

As it is evident from the above constraints, the goal of this algorithm and this simulation is to make exit strategies as realistic as possible. By grouping certain agents together for instance, we can achieve realistic movement of participating agents, as in reality, people usually move in groups. families, friends etc. Furthermore, congestion is a big part of the evacuation scenario, as it is congestion that determines how fast or efficient the evacuation of agents occur and also how it would take to move groups of people. With groups of people clumped together, the next most natural constraint would be to assign varying speeds as not all agents and not all groups of agents travel with the same speed across the topology of the building. There is also one other constraint which determines the cell capacity, which determines how spaced out and occupied agents are in a room, a hall or a passageway. This also help determine how many can get through a single doorway in order to transition from one passageway to another. This of course leads to previously mentioned issue of congestion.

All these constraints are hence taken into account as the simulation is performed in order to clock and test for the performance time and efficiency of the algorithm used for micro and macro model simulation.

1.5   Thesis Outline

The structure of the thesis is formulated as follows: chapter 2 provides a detailed literature and background work pertaining to the area of agent based simulation models and exit strategies. Chapter 3 focuses on the implementation details of the algorithm and specific scenario details that are incorporated in order to extend the algorithm in order to adapt it to a more realistic setting and also technical details regarding the PedSim simulation environment. Chapter 4 presents the results of the comparisons of the optimized(realistic) simulation between the macro and micro agent model setting and to chart out the various performance metrics that are obtained. Chapter 5 attempts to draw conclusions based on the obtained performance metrics and also present the future work related to the aforementioned. This thesis also provides a technical appendix in reference for the implementation of simulation scenarios within the environment of PedSim.

Chapter 2

Related Work

In this section the background literature and the work related to agent based simulation is dicussed.

2.1 Pre-Modern Disaster Preparedness, Management and Relief Techniques

Although disasters have existed throughout the history of the universe for all living organisms, human beings are perhaps the only one to be affected on a massive scale as a collective group as we are vulnerable and weak compared to other animals that live and co-exist with us. Hence human beings have taken many steps over the years fortifying and defending ourselves as best as we can through the various buildings and escape routes we design. The early inhabitants of mankind were not idle and did not become easy victims. There is historical evidence that the early man took various measures in order to cope with, reduce and mitigate risks. The mere fact that they chose to inhabit caves are a testament to this theory [12].

The ancient man also used other natural techniques for disaster preparedness. One of the prominent ones include the observation of animals since they are very perceptive

to natural disasters, animal behavior can and has been used for the better part of the millenia to predict the onset of a natural disaster such as floods, earthquakes etc. One of the earliest works to be done in this field involves in the observation of the behavior of fish, specifically the catfish, as they were found to exhibit a definitive behavior in advance of the occurence of an earthquake [20]. Other experiments and observations pertaining to such natural responses include the obeservation of ground electric field effects on behavior of Albino rats, Mongolian gerbils (sand rats), hair-footed Djungarian hamsters, guinea pigs, and red sparrows [21]. A summary of such animal behavior and work is presented by Neeti Bhargava et. al. [7].

To combat various natural disasters that affect us, mankind has tried for over a millenia to improve and adapt buildings, and entire cities to cope during the time of a catatrosphe. There are two broad categories to the mitigation of these disasters that have been long employed [13]:

- structural based mitigation

- non-structural based mitigation

Since my work pertains to indoor simulation and evacuation management, the discussion for non-structural mitigation forms here is considered unnecessary and beyond the scope of this thesis. Structural mitigation involves the presence of a building or a man-made mechanical or technological adaptations performed to reduce hazards and mitigate risks. The following sub-section expands more on the more modern techniques that have been developed over the course of the modern era and their implications. Although better and state of the art systems have been developed over the years, these systems have complications of their own. Failure of the perfect working of an evacuation system leads to the possible death of all the agents who are in need of evacuation.

2.2 Routing Algorithms and IoT Based Evacuation Management

Routing algorithms and maze solving algorithms have especially been applied to the structural domains for the real time analysis of shortest paths and obstacle avoidance by both machine and human agents since the time the earliest shortest paths algorithms were developed. The development of the shortest path algorithm by Dijkstra in 1956, especially has seen a plethora of applications in various fields. As one can guess most of its applications have been catered to graph based problems and obstacle avoidance. Shortest paths exit strategy formation approach is a good way to analyse a safe egress as it provides obstacle avoidance as well. How can it successfully be applied to a real disaster scenario? Before the advancements were made to IoT based infrastructure, maze solving algorithms were used to successfully analyze a safe egress.

IoT and computation systems have brought about a tremendous potential to perceive the environment and then form the necessary strategies that are required to safely escort the human agents to a safe exit point during the time of disaster. Modern evacuation systems have sensors in place to allow for better perception of the world during a disaster scenario. For instance the work by Kobes et al. determined that during fire related disaster scenarios, 56.3% of the participating agents were able to determine the exit points based on the exit signs when there was no smoke present whilst 81.8% was able to identify exit points only based on the exit signs when their vision was imparied due to smoke [22]. Based on such data for instance, it greatly enhances our need to analyse the fastest and shortest egress path so that all human agents in the vicinity can get to the exit points. The shortest path algorithm - Dijkstra's algorithm is currently used as a great tool to provide the shortest exit points as it is demonstrated in the work by Jehyun Cho [9]. This research pertains to the

dynamic analysis of a shortest path algorithm based on information obtained from the sensors and the smart infrastructure.

In order to address the issue of agent(s) and infrastructure mapping during a disaster event, smart sensors are placed all through the infrastructure in order to perceive the environment and the surrounding structures in order to analyze a safe egress. Motion detectors for instance are used to understand how many people are still trapped inside the building during a catastrophe. One example of using IoT technologies is proposed by Prasad Annadata et al. using multiple WIFI channels and the aforementioned motion detectors [4]. Based on the statistics gathered heuristic solutions are proposed in order to detect the number of personnel inside the infrastructure as well as perceive the environment.

Although the proposed methods and techniques are helpful in determining the shortest path to an exit point, they rarely can be used in a realistic scenario due to its simplistic nature. A real world scenario is more complex due the social dynamics of people. Such complexities are better understood while simulating agents in real and static simulations and then performing an analysis based on the observed stats. Such is achieveed through this thesis.

Existing literature on surveying personnel and the surrounding infrastructure for assistance based support for immediate first responders include the work done by Palmieri et al., who proposes a hybrid cloud architecture to manage and store the necessary required resources to command and control activities during emergencies [25]. However the work done by H. Muccini et al. [24] aims to improve this work by adapting geolocation of first responders to track people during a disaster for evacuation. Furthermore W.Choi et al. [10] proposes to model building evacuation by dynamic flow maximization and by considering variable capacities on some arcs as a function of flows in incident arcs. For the purposes of modeling our topology we have used a similar arc based geometry for determining the topological capacity of a

particular cell into which the entire infrastructure is divided into. Chen et al. [8] in addition to the above, proposes a flow control algorithm that calculates evacuation paths depending on building plan and total number of evacuees. Computation in this case aims at minimizing total evacuation time and assigning an optimal number of evacuees to each evacuation path. However [5] provides a robust solution by architecting an IoT system to monitor and update dynamically the topology of the environment.

Another important aspect that remains unaddressed is that in real disaster scenarios, over crowding and congestion of certain pathways and exit points are bound to occur frequently. The issue of congestion is serious and perilous and is arguably pointed out by a case study performed by John El Khoury [16] and analyses the dangers due to high traffic intensity during a disaster event. Although the analysis is performed on a wide scale city range, the implications of congestion and other such constraints are just applicant to a much smaller scale infrastructure. To dynamically reallocate personnel to routes that are not just shortest paths but optimal paths is what we try to achieve through these simulations and added constraints. The work done by Antoine Desmet et al. [15] addresses the congestion issue by a "self adaptation" algorithm much inspired by the computer network routing algorithm - The Cognitive Packet routing algorithm by E. Gelenbe [17]. A robust evacuation with optimized routing solution proposed by H.Muccini et al. [5] are as follows:

- Optimal solutions that can be continuously updated, so evacuation guidelines can be adjusted according to visitors position that evolve over time.

- Paths that become suddenly unfeasible can automatically be discarded by the system.

- The model can be incorporated into a mobile app supporting emergency units to evacuate closed or open spaces.

The aforementioned algorithm although is complete and optimized, additional dynamic and realistic constraints are yet to be added during an evacuation scenario. The constraints are 4 fold as mentioned in the introduction section. Over the course of chapter 3, the details of the algorithm is explained and how these constraints provide a more optimized simulation scenario since it considers congestion, grouping, varying age and speeds in addition.

2.3   Simulation Tools and Constraints

In order to accomodate for accurate results and model precise human behavior, agent based modeling and simulations have become increasingly popular as computational expenses became cheaper and ever more accessible. In addition to developing state of the art routing algorithms and other obstacle avoidance mechanisms, it was also absolutely important to understand human behavior and the complex decisions that agents make during a catastrophic event. To understand the implications of human behavior and complex real-time decision making and other realistic constraints, this thesis presents various scenarios that are depicted in a small to a large scale microscopic agent based fashion in hopes to draw out some interesting conclusions based on the obtained results. One such example of the kind of work that we simulate is the impairment of certain agents based on the circumstances that are met with in real time during the disaster event. The work presented by Selain Kasereka et al. [18] offers a unique insight into modeling agents for a fire disaster and simulating the case for some interesting results. It should be mentioned that the case presented in the current thesis however is modeled for a generic scenario. In the work presented, the smoke and fire can affect the agent, thus reducing their potency and their ability to escape from danger. This damage to potency can be essentially translated to a re-

duced movement speed and a severe impairment to their cognitive ability to perform real time decisions.

Another really interesting thing to point out is about psychology and human behavior during ground zero scenarios. Human panic and confusion is generally a chaotic element that must certainly be infused into such simulations as they provide further insight into human behavior and particularly decisions that lead to increase or the decrease in congestion. Based on certain decisions and paths, it may cause discomfort or physical pain to some and many agents within the promixity of such an incident. Ashutosh Trivedi et al. [27] provides a detailed analysis and where different strategies are evaluated and the corresponding evacuation time and physical discomfort caused to the agents are observed. He justifies the effect of social forces based on cohesion which, in my thesis, is also considered whilst perfoming simulation for microscopic agents. Our work revolves very closely the work done by A. Trivedi et al. as it also deals with social forces, panic and other realistic constraints. Through this thesis, experimental analysis for these sophisticated constraints are performed for both macro and micro agents in hopes for better understanding and formulating a safe egress for the agents.

Simulation tools have been used since the inception of computers and computation technology since the early 1970s. Modern simulations for evacuation and relief management includes the simulations of pedastrians and agents within the domains of the inscribed topology, with start and end points. These simulations are then run against various parameters and scenarios thus obtaining various results. Through this thesis the various experiments are simulated within the environment of PedSim, a massive microscopic real time simulator. The in depth analysis of large number agents based simulators are provided by the extensive and comprehensive survey presented in "Agent Based Modelling and Simulation tools: A review of the state-of-art

software" by Sameera Abar et al. [3]. The survey is based on the following criteria which is presented as follows:

- License/Availability

- Source Code in which the simulator is written in

- Type of Agent based on its interaction and behavior

- Coding language or application programming interface(API) for Model Development; Integrated Development Environment(IDE) used

- Compiler/Operating System/Implementation Platform

- Model Development Effort

- Modelling Strength/Simulation Models' Scalability Level

- ABMS scope or Application Domain

From the extensive survey provided, it was easily inferred that PedSim, pedastrian microscopic simulator was the best suited for testing the various scenarios as it could support both micro and macro based agent modelling scenarios and grouping. It must also be mentioned that the tool is entirely written in C++, extremely light-weight and highly scalable. It also works well on development platforms and can be compliled easily on all operating systems. Its simple and easy to use, as complex simulations can be run for massive number of agents and the results can be visualized in real time. For all these reasons I have decided to use PedSim exclusively as my test bed tool whilst modelling such scenarios.

In the following section the scenario and the implementation details of my work will be discussed.

Chapter 3

PedSim Environment and Setup

This section briefly explains the underlying PedSim library modules and the basic algorithm that is presented in [24], based on which more constraints were added for realism and optimization; used for the simulation of pedastrian agents. The following subsections will describe the environment and the topology of the building, the algorithm that is used for the movement of participating human agents and the necessary changes that were made to make the algorithm more realistic and the technical implementation details of the PedSim simulation environment.

3.1 Scenario Description

The work involves simulation of level 3 of Coppito 0 building during a disaster scenario such as a fire. The building topology consists of 26 rooms and cubicles connected by several hallways and pathways for exit and is depicted in 3.1. There are four exits in total. Two in the middle top and bottom denoted by a the red font as **E** and **E"**. There are two more on the left and right which is depicted by **E'** and **E"'** also in red. The entire premises is divided into cells which form the basic unit of analysis. Each room is made up of one or more cells. For instance the room at the top left corner is made up of cells 96 and 97. Dark blue lines indicate wall boundaries. Each cell for

our scenario has a dimension of 4.5x3 in meters. The doorways provide a separation of 1 meter each. Roughly at the center of the building, two conference rooms and a hallway provides a substantial gathering area.

The work here caters to simulating the flow of crowd during the event of a fire across the four exits. Restrictions such as exit clogging due to intense smoke/fire can be imposed redirecting most of the traffic across specific other exits. Also crowds from crowded exits can be siphoned off by redirecting them to an alternative less crowded exit. The data flow density of population which used for simulation in Pedsim is measured using an IoT framework of smartphones, cameras, RFIDs and motion sensors that enable localization of victims in the room. The tool was thus used as a framework to simulate evacuation algorithms in Coppito 0.

The given figure 3.1 is the graphical description of Coppito 0 building and is referred from the outline and scenario as stated in [24].
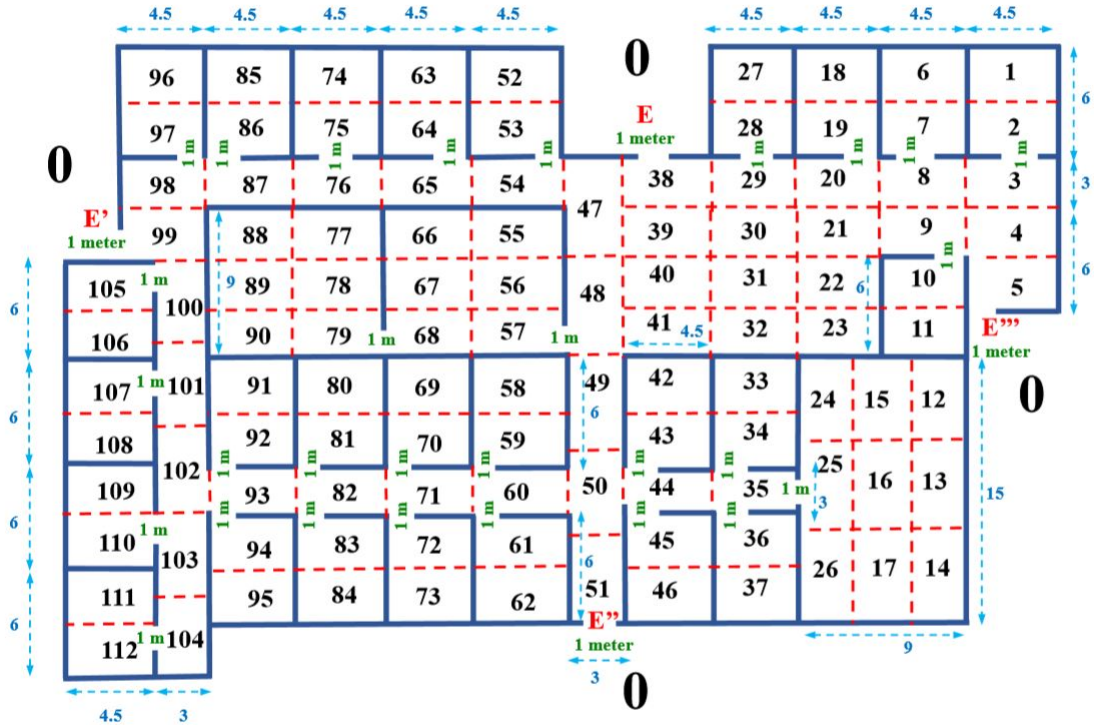
Figure 3.1: Coppito 0 Topological Layout

A detailed description of adaptation of the tool to suit the given scenario is described in the next section.

3.2   PedSim Library Modules

The Pedism library allows for the use of pedestrian dynamics into our own software. The libpedsim simulation rendering engine can be extended, modified and modeled to suit specific behavior patterns and scenarios. The above disaster considered is fire and through this thesis, we aim to demonstrate the capabilities of this microscopic simulator, modeling crowds during an emergency evacuation using the above mentioned building topology.

The implementation of the various scenarios(which will be explained briefly in the following section) using Pedsim is developed on Ubuntu 18.04 using libpedsim version 2.4.2. The library itself contains many subsections and modules which handle specific tasks such as agent movement, topology description etc. The various modules of the pedsim library is described below.

The libpedsim tool can be broadly classified into 2 sections - the simulation of pedestrians and graphically rendering the simulation process onto a QT based graphical window to depict the flow of agents in real time process.

The following modules are part of rendering output to a graphical window:

1. agent.h

2. agent.cpp

3. cell.h

4. cell.cpp

5. config.h

6. config.cpp

7. control.h

8. control.cpp

9. control.ui

10. grid.h

11. grid.cpp

12. mainwindow.h

13. mainwindow.cpp

14. moc_loadscene.cpp

15. moc_control.cpp

16. moc_mainwindow.cpp

17. moc_scene.cpp

18. moc_predefs.cpp

19. qrc_application.cpp

20. scene.h

21. scene.cpp

22. style.h

23. style.cpp

24. tree.h

25. tree.cpp

26. ui_control.h

As the above modules are used to render the algorithm to a graphical output, most of the modules as mentioned above need very little to no modification whatsoever. It is also worth noting that the modules agent.h and agent.cpp contains the actual definitions for the behavior of pedestrian agents. These behavior functions are implemented by the author of the tool as a generic inter-agent based interaction and is developed keeping in mind to add further and more complex behavioral functionalities. These functions can be extended in the ped_agent.h and ped_agent.cpp respectively. The following modules below represent the core modules that we extend and modify to suit the scenario at hand:

1. coppito.h

2. coppito.cpp

3. loadscene.h

4. loadscene.cpp

5. ped_agent.h

6. ped_agent.cpp

7. main.cpp

The first two modules - coppito.h and coppito.cpp is an external module that is incorporated into the source of the libpedsim library for the specific purposes of modeling the building Coppito 0, as the name suggests. The detailed description of coppito.h and coppito.cpp will be listed after the brief description of the other mentioned modules.

loadscene module is used to generate and extract the specific topology of the building. The exact dimensions of the building is stored on an .xml file which is then

fed into the loadscene module to incorporate into the library source for graphical representation. This .xml file has specific tags which can be used to not only describe the dimensions of the building but also to specify the number of agents and the path of their trajectory - mentioned as waypoint within the .xml file. This information is then retrieved by the loadscene module, extract information from the various mentioned tags, and generating the mentioned number of total agens on the graphical output, adding the constrained waypoints to the agents, and most importantly, to gather the information required to generate a graph that depicts the nature of the building in description.

The ped_agent module describes the behavior and movement of the agent that is to be rendered to the graphical window. This module consists of behavior functionalities that typically incorporates social forces, obstacle forces, look ahead force, desired force and my force. The definition of these forces (generic navigation constraints and behaviors) are explained below:

1. my force:

   - myForce() is a method that returns an "empty" force (all components set to 0). This method can be overridden in order to define own forces. This can thus be used to model more complex human navigation/decision making patterns.

2. lookahead force:

   - This calculates the mental layer force of the strategy "look ahead". It is implemented here in the physical layer because of performance reasons.

3. obstacle force:

   - This calculates the force between this agent and the nearest obstacle in this scene.

- It iterates over all obstacles.

- Hence the complexity of this module is equal to $O(N)$.

4. social force:

   - This module calculates the social force between this agent and all the other agents belonging to the same scene. It iterates over all agents inside the scene.

   - Hence it has a complexity of $O(N^2)$.

5. desired force:

   - This module calculates the force between this agent and the next assigned waypoint. If the waypoint has been reached, the next waypoint in the list will be selected.

   - At the moment, a visited waypoint is pushed back to the end of the list, which means that the agents will visit all the waypoints over and over again.

The main behavioral functionalities to be incorporated is thus implemented in the ped_agent module. The agents are made to move according the complex grid geometry based architecture called *Alan Turing Building Architecture* which is implemented in the coppito source module.

As aforementioned, the .xml file contains a generic description of the topology of the building that is to be modeled and used for rendering simulation. The coppito.xml is the file used in our case for the modeling of our present scenario. This information is retrieved by the loadscene module to extract the exact graphical plot points to draw the building.

The following figure 3.2 depicts the environment of PedSim and the coppito 0 building that is modeled.
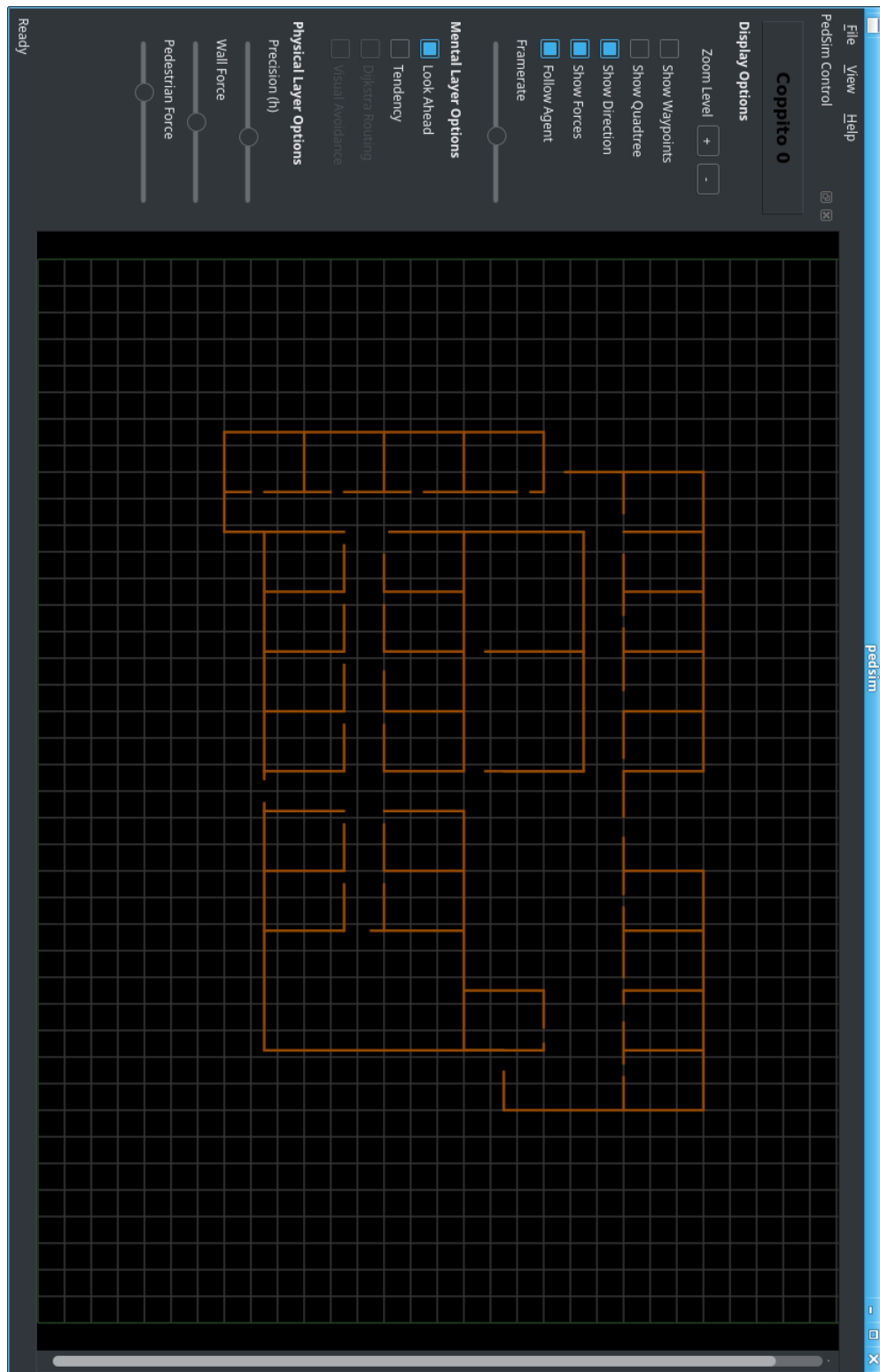
Figure 3.2: Grid Representation of Coppito 0 within the PedSim Environment
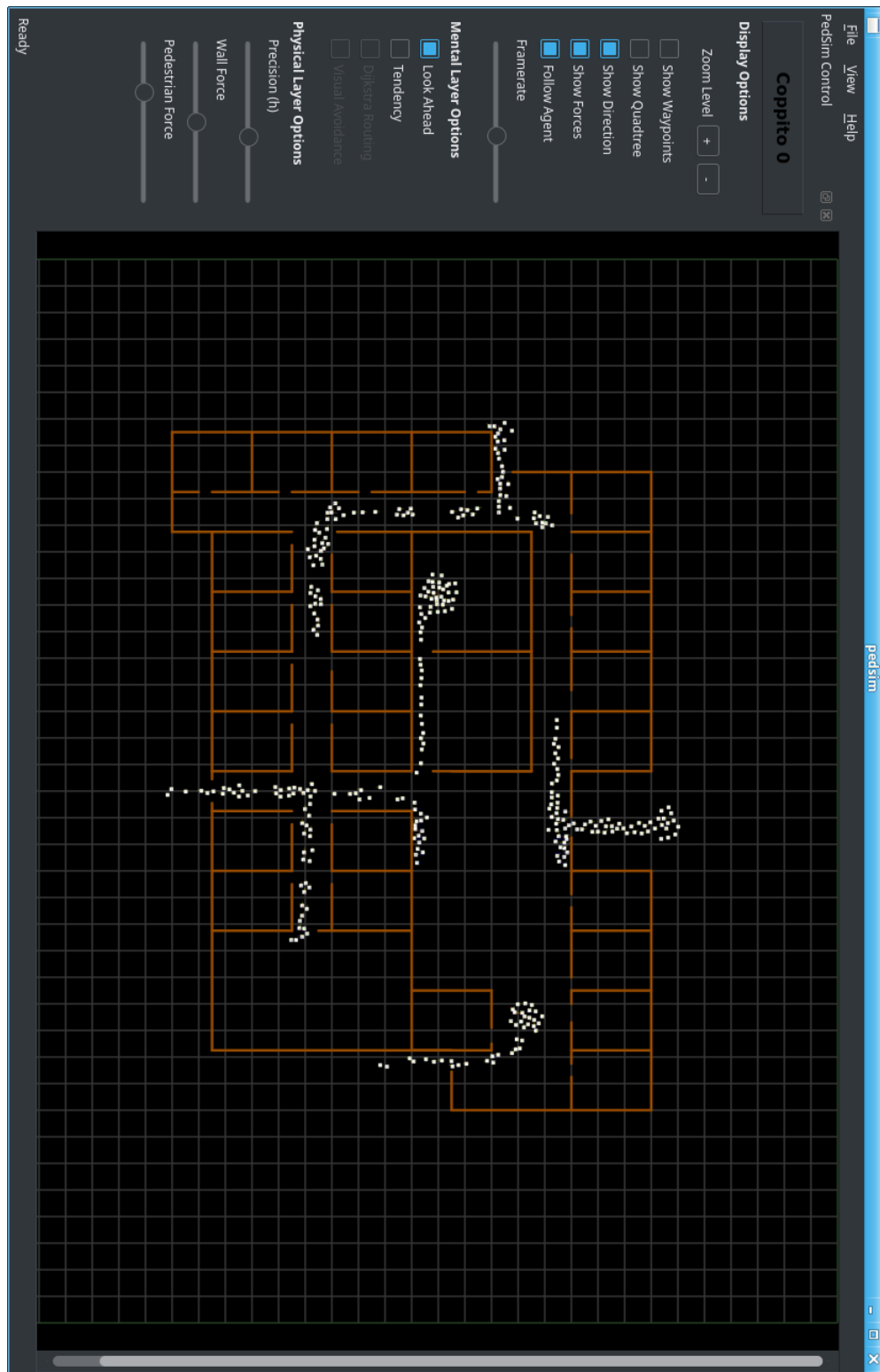
Figure 3.3: Microscopic Agent Simulation in PedSim

The above figure 3.3 represents the microscopic agents being simulated within the PedSim environment, as the agents are subjected to the various constraints and are tasked to evacuate to the safe zone "**E**".

PedSim also provides a quick and easy way to change certain key constraints within its front end QT client. The following figure 3.4 depicts the variables that can be changed in real time whilst the simulation is being processed.
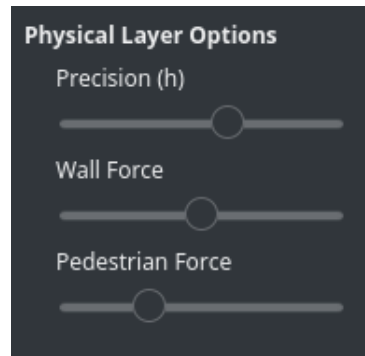
Figure 3.4: Real Time Variable Constraints in PedSim

The above figure depicts 3 variable sliders that can be applied to the simulation in real time. The three sliders are explained as follows:

1. precision(h) - this represents the time steps τ. These are discretized time steps that are used to calculate the total time taken for the evacuation of the participating agents.

2. Wall Force - this force determines how close to the obstacles they can tread, this is extremely useful in modelling situations like fire, heat etc., in cases where the walls of the topology should not approached due the nature of the disaster or situation.

3. Pedastrian Force - This force essentially determines the social forces among the participating agents. The more the pedastrian force the less intermixing

happens between the agents. This variable constraint is especially important in modelling cell capacities and in determining flow dynamics.

The following subsection describes in detail the algorithm that is used in the simulation of the various scenarios.

3.3   Algorithm Description

All the agents within the PedSim environment are modeled and simulated according to certain realistic constraints. These constraints form the backbone that defines how the agents interact with one another and the environment that surrounds them. To make things simpler, the modeled topology is subdivided into grids. These grids are further divided into atomic square divisions of certain length and breadth. For convenience, we term these square divisions as cells. Constraints such as capacity, flow etc. are imposed on these cells to create a flow dynamic that is closer to the realistic simulation of an emergency evacuation. Within the premise of the running program, these are plot points which are manipulated by the program.

The coppito module pipelines this plot information for further processing. The basic movement of agents are strictly modeled according to the algorithm presented in [24]. Various constraints are mentioned in the paper for the anaylsis of a shortest egress path during design time. However these constraints alone are not enough to make the simulations close to be realistic. Once the basic algorithm is explained, further conditions will be introduced and explained.

In the work by H.Muccini et. al [24], discussion of the linearization of the constraints for effectively reducing the evacuation time of the agents is mentioned. However since its redundant to use linearization within the pedsim framework to simulate for the scenario, we mainly consider only 3 of the below mentioned constraints.

$$y_j^t - y_j^{t-1} - \sum_{i:ij\in A} x_{ij}^{t-1} + \sum_{i:ji\in A} x_{ji}^{t-1} = 0 \qquad (3.1)$$

$$j \in V, t \in T, t > 0$$

$$0 \le x_{ij}^t + x_{ji}^t \le c_{ij} \qquad (3.2)$$

$$t \in T, ij \in A$$

$$0 \le y_i^t \le n_i \qquad (3.3)$$

$$t \in T, i \in V$$

Where,

1. $T = \{0, 1, ..., \tau\}$, set of unit of time slots.

2. y_i^t = state of cell $i \in V$ at time $t \in T$, that is, the number of persons that occupy i at t.

3. n_i = capacity of cell i; it measures the maximum nominal amount of people that i can host at any time.

4. x_{ij}^t = how many persons move from cell i to an adjacent cell j in $(t, t+1]$; this gives the average speed at which the agent flow proceeds from cell i to cell j.

5. c_{ij} = capacity of the passage between cell i and cell j; this is the maximum amount of people that, independently on how many persons there are in cell j, can traverse the passage in the time unit.

The above conditions are a set of mandatory inclusive pre-requisites required to be included from *An IoT Software Architecture for an Evacuable Building Architecture* by H. Muccini et. al [24], since the experimentation of various scenarios are based on

the aforementioned set of equations and conditions. Through this thesis we introduce 4 more constraints to be included along with this base algorithm. The specific nature and details of the additional constraints will be discussed in the next chapter, where the various different experiments and scenarios are carried and are analyzed. It must also be mentioned that in order to be coherent with the mentioned work and also to analyze optimal egress paths the following condition is also considered in addition to the above 3:

$$\max y_0^\tau \tag{3.4}$$

The above condition is obviously included in order to maximize the number of evacuees during the disaster scenario.

3.4   Topology Formulation

The topology of the building as given in 3.1 is modeled according to the *Alan Turing Building Architecture* and as shown in the figure the whole Coppito 0 building is subdivided into cells of equal length and breadth. Each cell also has a cell capacity as defined above and is assumed that inter-connected cells have full passage, if not bound by walls or walls with reduced passage capacity(doors).

The coppito source module divides the entire topology into such grids that are composed of cells, whose initial plot point is taken from the loadscene module, which in turn takes its information from the coppito.xml file. To translate simple plot points to a complex cell based grid structure, a highly complex user defined data structure was developed to store this information.

The plot points that are stored into the .xml file is randomly plotted with the exception of the first starting point in order to facilitate a quicker render of the entire building. However for the processing of a cell based structure, this information

cannot be random and hence the information is then once again converted to form a structured cell geometry.

Thus a complex user defined data structure is instantiated in order to hold all this information in place. For the sake of convenience and optimal storage of data, the coppito 0 building is divided into regions called blocks. Each block contains a set number of cells and naturally have a variable number of cells. The whole topology is divided into 18 blocks and consists in total of 119 cells. Each cell is also of a specific dimension and has a certain cell length and a cell width. To understand what a block is, it can be easily explained diagrammatically as follows:
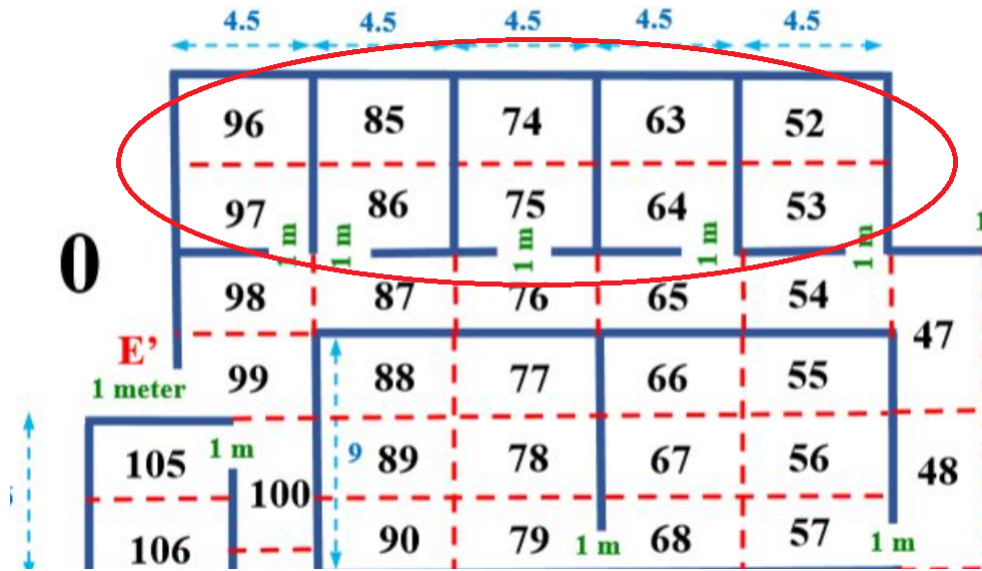
Figure 3.5: Groups of Cells Making a Block Structure

From the above figure 3.5, the red encircled circled area which includes the cells numbered 96, 85, 74, 63, 52, 97, 86, 75, 64, 53 form a block. Although a certain number of cells make up a block, the number of cells that make a block is often variable. Essentially one section of the topology is considered a block. This helps segregate different areas of the premises and identify them easily.

Further decomposing the cell, subdivides it into 4 edge lines, each edge line further decomposing into 4 vertices, an end to end x and y plot points. A flag is used to

30

identify if the edge lines are walls and also if they incorporate doors. This is highly useful for creating a tree structure pattern for path finding, used by agents during evacuation simulation. The final complex structure that is defined to hold this cell structure is explained in detail in the Appendix section. In order to dynamically store the data of the cell information, the data structure used in place is a custom user defined 7 layered structure.

There are several sub-modules implemented in this custom made source file. The following are methods that are implemented as part of the module:

1. divide_cells

2. cell_structure_allocation

3. vertical_cell_allocation

4. non_standard_vertical_allocation

5. wall_allocation

6. wall_division_horizontal

7. wall_division_vertical

8. door_allocation

9. graph_tree_structuring

10. print_block

11. print_walls

12. print_door

13. print_graph

Further details regarding the technical and programmatic implementation of these modules are provided in the Appendix section B of the thesis. The following section discusses the modification of the existing unoptimized simulation scenario from [24]. The different simulation results comparing macro and micro agent scenarios are presented and analyzed. The resulting implications of these observations are discussed.

Chapter 4

Experimentation and Analysis

This chapter is intended to provide the experimental proof of the proposed extension of the previously formulated algorithm from [24]. Through this chapter, the detailed explanation of the additional constraints that are added to the unoptimized simulation from the previous chapter is provided. Next micro and macro agent simulations are presented pertaining to certain cases and scenarios to compare between the unoptimized and the optimized simulations. The resulting observations and statistics are then presented for further analysis and the implications are then discussed.

4.1 Optimization through Constraints

From the equations 3.1, 3.2 and 3.3, the basic conditions for agent occupancy, flow control and cell capacity is discussed. However these conditions are hardly realistic when compared to modelling a real crowd of agents. In realistic scenarios, groups of people have altered behavior, confusion panic etc. There will be different forms of social constraints, varying movement speeds, and due to the impaired cognition, perceptions can change leading to different decisions that are made during a catastrophe. To model some of these realistic scenarios, it became evident that the inclusion

of certain more properties to the scenario was mandatory. The following conditions as mentioned in section 1.4 is analyzed here:

1. Cell capacity specification by defining social distances

   - Although PedSim comes by default with a cell class, it does not go coherently with the topology, as cell divisions are highly subjective and can change their length and breadth according to the defined scenario. To cater for this, manipulation of the social forces constraint causes agents to maintain a certain "social distance" from each other. This can be used to our advantage to simulate cell capacities based on these social distances.

2. Definition of total area capacity and doors flow capacity constraints -congestion control

   - The simulation present [24], unfortunately does not present the congestion scenario as well due to the simplistic constraints present. Providing additional constraint for area capacity, door flow capacity and passageway capacity can help reduce congestion and enable the re-routing of agents to alternate exits or routes in real time.

3. Simulating social attachment among some agents

   - With the introduction of the group constraint, social attachments can be modeled for a more realistic approach. For instance friends move together, a mother will most likely not be separable from her child etc. These groups can be defined during the simulation and then observed for the various real time decisions that these groups of agents make.

4. Setting the speed accordingly for various groups

- By default PedSim models for microscopic agents and hence the entirety of the agents are considered as a single group. The movement speed for these agents are varied across a distribution and the average speed set for the microscopic group of agents. By setting a variable speed constraint and with the introduction of groups, we model PedSim to cater for macro and micro agent simulation and thus support varying speeds according to the different group formations that can be defined during the simulation. In other words v_{max} is not fixed across all agents.

The PedSim library is extended to be inclusive of the algorithm from the section 3.3 and the aforementioned constraints in order to simulate the various scenarios for micro and macro agents. The following section provides a detailed analysis of the various scenarios that are simulated within the PedSim environment.

4.2   Simulation and Scenario Analysis

This section contains the experimental analysis based on the scenarios that are presented in the forthcoming subsections. It must be noted that the following simulations are run on a PC with the following specifications - Ubuntu 18.04, 8 GB ram, and an i5 processor with 2.5 ghz clock speed. A series of sub-sections are present in this section - each representing a particular scenario for simulation. The scenario will be described and the corresponding tabular observations and graph is then presented.

There are some properties that are common to scenarios. We follow similar topological constraints as given in [24]. For instance, in the previous work presented, the cells are assumed to be isometric, i.e. the cells are bi-directional and can be crossed from any direction with the same amount of time. In the scenarios presented, the cells are assumed to be isometric as well.

According to the work by Daamen et al. [14], door capacities are presented, where they are based on varying stress and composition levels of agents and he subesquently proposes an average of 2.8 persons per second for a 1 meter wide door ($p/m/s$). In order to perform a statistical analysis and to remain coherent with the results with from [24], the door capacities and cell capacities are modeled after the presented stats in the aforementioned work. In [24] the pessimistic and optimistic values considered are 1.03 $p/m/s$ and 3.23 $p/m/s$ respectively. This translates to a maximum of 5 people in pessimistic and a maximum of 16 people in optimistic situation can pass through a 1 meter wide door per slot time (5 seconds). Hence the same values will be considered here as well. There are also a certain terms that are used in this section, namely, optimistic and pessimistic. Optimistic scenario assumes that no passageways and exits are blocked. Whereas pessimistic scenario assumes that 50% of the emergency exits are blocked and passageways can accommodate only 5 people at a time.

The maximum number of people considered is 1008, as mentioned in [24]. All the above conditions are compliant with the mentioned paper.

4.2.1 Case 1: Macro Agent Simulation Comparing Optimized and Un-Optimized Simulations

The first case is depicts the CPU run time comparison between macro agents(single or no group concept is present here) that follow the optimized algorithm from [24] against the proposed extension of the simulation by additional constraints. For the current scenario, pessimistic path scenario is incorporated in order to keep the coherency of results and perform the necessary comparisons with the aforementioned paper.

A total of 1008 people are taken for the purposes of this simulation. Various walking speed scenarios among acquintances and friendly dyads are presented in the work by Wagnild et al. [28], however since there are no complex group formations

that are currently part of the simulation, the speed of 1.2 m/s estimated in J. Ye's average *free flowing walking velocity* [29] is used here.

The original code for simulation was written on OPL language and solved on CPLEX version 12.8.0 [24]. However codebase for the modified algorithm has since been translated to the PedSim environment. The following table 4.1 represents the simulation times between the optimized and the un-optimized modeling scenarios.

	Evacuees	CPU Time (sec) - Optimized	CPU Time (sec) - Un-Optimized		Evacuees	CPU Time (sec) - Optimized	CPU Time (sec) - Un-Optimized
1	20	0.25	0.28	27	540	4.9	2.61
2	40	0.28	0.31	28	560	5.1	2.77
3	60	0.47	0.44	29	580	5.7	2.84
4	80	0.61	0.53	30	600	6	2.96
5	100	0.63	0.47	31	620	6.5	3.1
6	120	0.69	0.58	32	640	6.94	3.53
7	140	0.72	0.6	33	660	7.2	3.32
8	160	0.75	0.61	34	680	7.76	3.54
9	180	0.82	0.71	35	700	7.96	3.91
10	200	0.88	0.76	36	720	8.4	3.42
11	220	1.1	0.83	37	740	8.7	4.14
12	240	1.22	0.88	38	760	9.3	4.16
13	260	1.27	1.01	39	780	9.88	4.17
14	280	1.33	1.09	40	800	10.4	4.19
15	300	1.4	1.12	41	820	10.79	4.3
16	320	1.6	1.44	42	840	11.5	5.13
17	340	1.82	1.28	43	860	12.37	5.07
18	360	1.99	1.33	44	880	13.01	5.12
19	380	2.1	1.57	45	900	13.68	5.27
20	400	2.3	1.61	46	920	14.5	5.36
21	420	2.44	1.73	47	940	15.4	5.49
22	440	2.6	1.88	48	960	16.04	6.01
23	460	2.9	2.02	49	980	17.07	6.35
24	480	3.3	2.08	50	1000	18.2	6.25
25	500	3.8	2.19	51	1008	19.4	6.47
26	520	4.4	2.35				

Table 4.1: Macro Agent Simulation - Optimized vs UnOptimized Simulation

It is evident from the above figure that the optimized algorithm takes an exponential time to compute the simulation times for the macro agents at the given number of time steps. Below is a graphical description of the observed statistical values from the table 4.1.
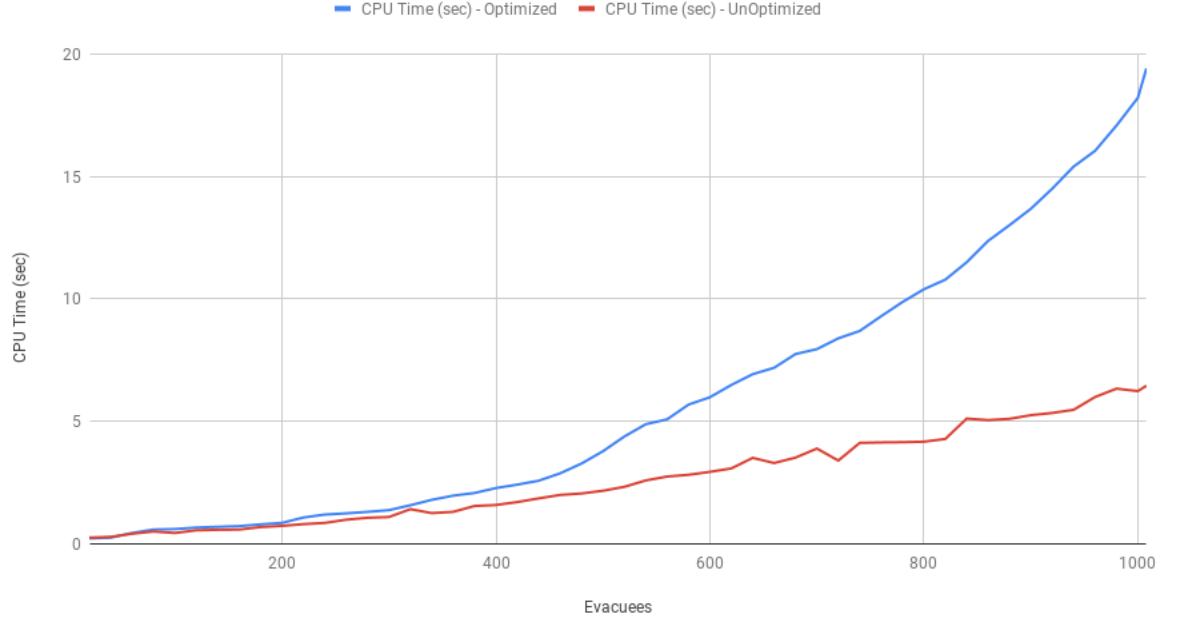

Figure 4.1: Macro Agent Simulation - Optimized vs UnOptimized Simulation

The above figure 4.1 depicts the difference in the performance experienced as the constraints placed are higher in count and complexity. In the present scenario, not all the constraints are incorporated however. In macro agent simulation, no grouping is employed, hence there is no grouping constraint. This in turn also affects the varying speed constraint as groups of agents are not involved. It is also noteworthy mentioning that single agents can be considered as individual groups as well. Since there is no such constraint, the average of 1.2 m/s velocity for all agents is applied and the simulation of the agents are then performed.

4.2.2 Case 2: Run Time Comparison between Micro and Macro Agent Simulation

This case is similar in terms of objective to the previous case 4.2.1. However the key difference lies in what is compared in this particular scenario. Micro and macro agent simulation scenarios are compared in this case. Since there is micro agents that are also included, grouping constraints are included and subsequently varying velocities based on grouping and affiliation.

Statistical Purview of Gait Speeds Among People

To understand the how human walking speed impacts the simulation performance and the overall evacuation time, the work presented by Simonsick et al. [26] is used as a baseline agent walking speed reference. The work presents an extensive statistical review on the walking speeds of people across all ages and sex. The walking speeds of people, according to Simonsick et al., is based on a variety of factors. These factors include not only a generic health purview and disabilities, but other social and psychological factors such as education and lifestyle. The factors that are taken into account are age, sex, education, marital status, smoker/non-smoking lifestyle, comorbidity, subjective health, waist circumference, height, neuroticism, extraversion openness, agreeableness, conscientiousness, systolic and diastolic BP, cholesterol levels, triglyceride levels, handgrip strength, averge walking activity per week etc.
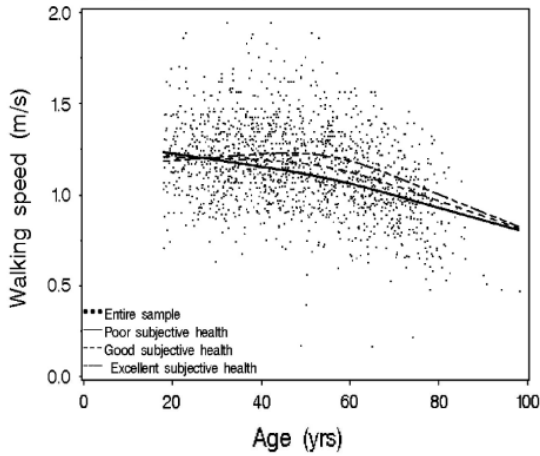
The following statistical graphs were obtained based on their analysis and observations as given in [26]:
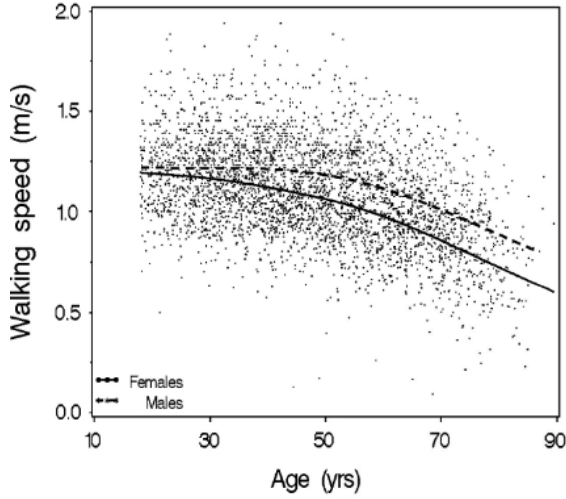
(a) fig a
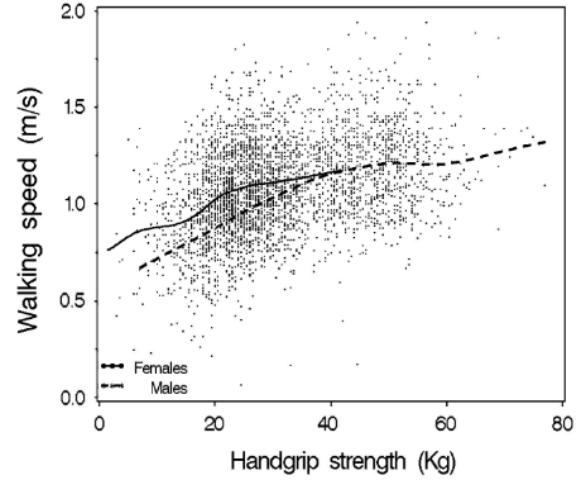
(b) fig b

(c) fig c
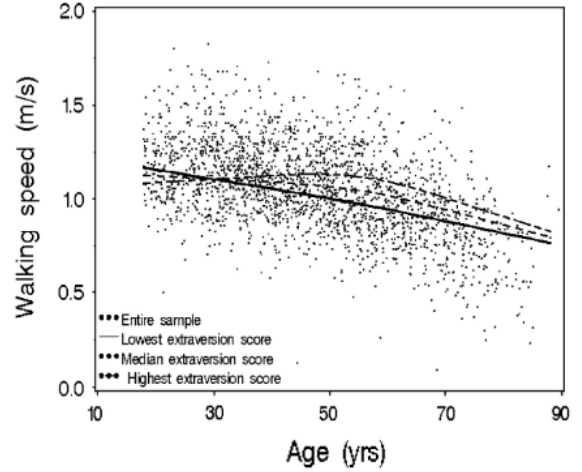
(d) fig d

Figure 4.2: Walking Speed Statistics

The above figure presents the walking speeds of people based on a few of the aforementioned factors. According to the study, the age of 55 was considered as the inflection point for walking speed sensitivity. All the statistical models hence were modeled considering 55 as the age of inflection. Generally the study found out that women walked slower than men at all ages, with more pronounced variations after the age of 55. Younger women walked faster than the older women. According to the study, the walking speed varied from 0.7 to 1.2 m/s.

Another study was performed by Wagnild et al. [28], and he determined that walking speeds are highly depend on the company and the lowest speed of the person present in the group. This scenario is inclusive of group constraints. However this scenario is also only to compare the actual running times of the simulated output. Complexities regarding group dynamics are not compared here. To keep the results coherent, the macro agents are considered with the uniform speed as mentioned in the previous section. The micro agents consists of mixed groups and individuals.

To present the impact of walking speeds in an evacuation scenario, the following graph depicts the evacuation time required in the case of variation of walking speeds of people who move slower than the average speed during an evacuation. In this scenario however, macra agent model is considered. The simulation trial is run with the same optimized algorithm from [24]. The total number of agents considered in the following scenario is $N = 1008$.
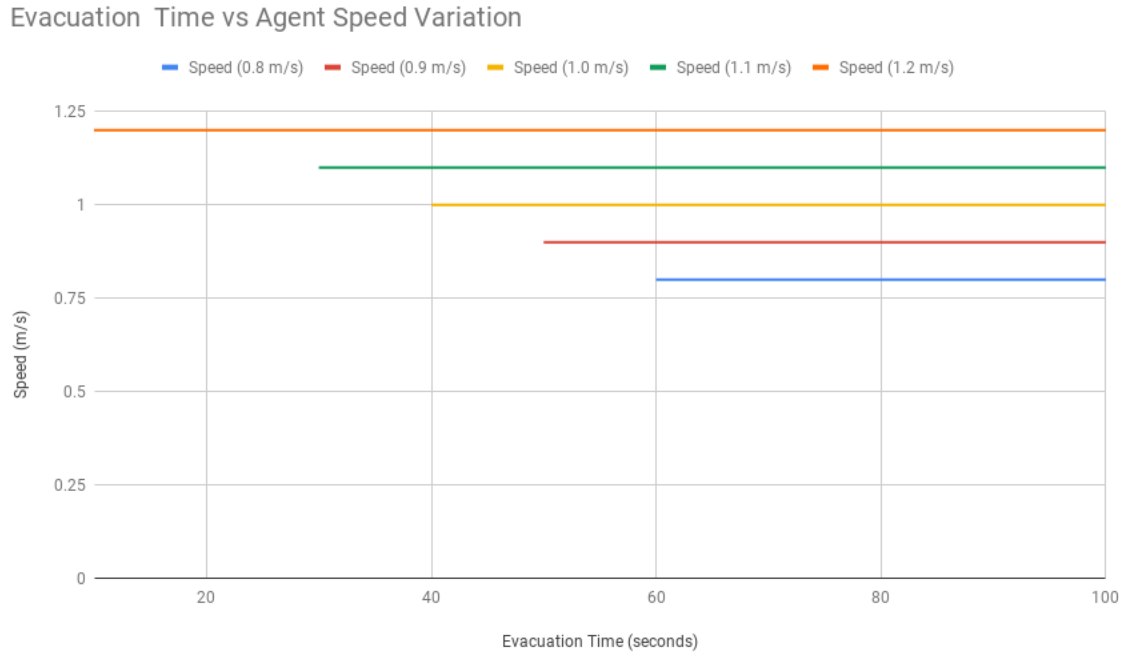

Figure 4.3: Evacuation Time vs Agent Speed Variation

In the given figure 4.3, it is explicitly shown how walking speeds directly affect the total evacuation time of the agents during a disaster event.

Computation Time - Micro vs Macro Agent Simulation

The comparison values are based off on the pessimistic path scenario as presented in [24]. A cross way comparison between macro agents with un-optimized simulation constraints, macro agents simulated with optimized constraints and micro agent with constraints is tabulated in the table below. The CPU run time values for the microscopic agents that are simulated using the un-optimzed are obtained from the charted values from table 2.0 from the aforementioned paper, based on which the other two comparisons can be made and analyzed. To keep group velocity diversity among the micro agents, 20% of the total number of micro agents are randomly distributed with varying speeds between the range of 0.7 m/s and 1.2 m/s.

A total number of evacuees upto 1008 are considered for the comparisons, similar to the previous section 4.2.1.

As it can be noted from the above table 4.2, the time taken for the micro agent simulation far exceeds the time taken by both the previous simulations - for macro agent model in both optimized and un-optimized simulations. However, it should be noted that the micro agent simulation has 20% of the entire participants at a reduced movement speed, during each simulation run. Since the number of reduced movement speed agents increase exponentially for each trial run, so does the computation time. These agents who have reduced speeds are also randomized to form certain affliate bonds and groups.

More discussion on the nature of such groups and their implications are discussed in the next sub-section where the evacuation time is discussed in detail and simulations are run in order to observe the actual time taken due to the additional constraints.

Evacuees	CPU Time (sec) – Macro Agent Sim:Optimized with Constraints	Evacuees	CPU Time (sec) – Micro Agent Sim:Optimized with Constraints
20	0.4	540	10.8
40	0.46	560	11.5
60	0.6	580	12.3
80	0.8	600	13.8
100	0.9	620	14.97
120	1.01	640	15.92
140	1.11	660	17.3
160	1.4	680	19.2
180	1.7	700	21.84
200	2	720	23.7
220	2.3	740	25.9
240	2.4	760	27.9
260	2.7	780	30.2
280	3.2	800	33.5
300	3.5	820	36.7
320	3.9	840	39
340	4.3	860	43.3
360	4.8	880	47.2
380	5.4	900	50.34
400	5.9	920	54.2
420	6.5	940	58.99
440	6.97	960	64.2
460	7.6	980	68.2
480	8.1	1000	74.78
500	8.7	1008	76.34
520	9.7		

Table 4.2: CPU Time Comparison between Micro and Macro Agent Simulation

The below figure 4.4 represents the comparison between the 3 major simulation models and their exceedingly apparent ranges of computation times.

Micro vs Macro Agent Simulation
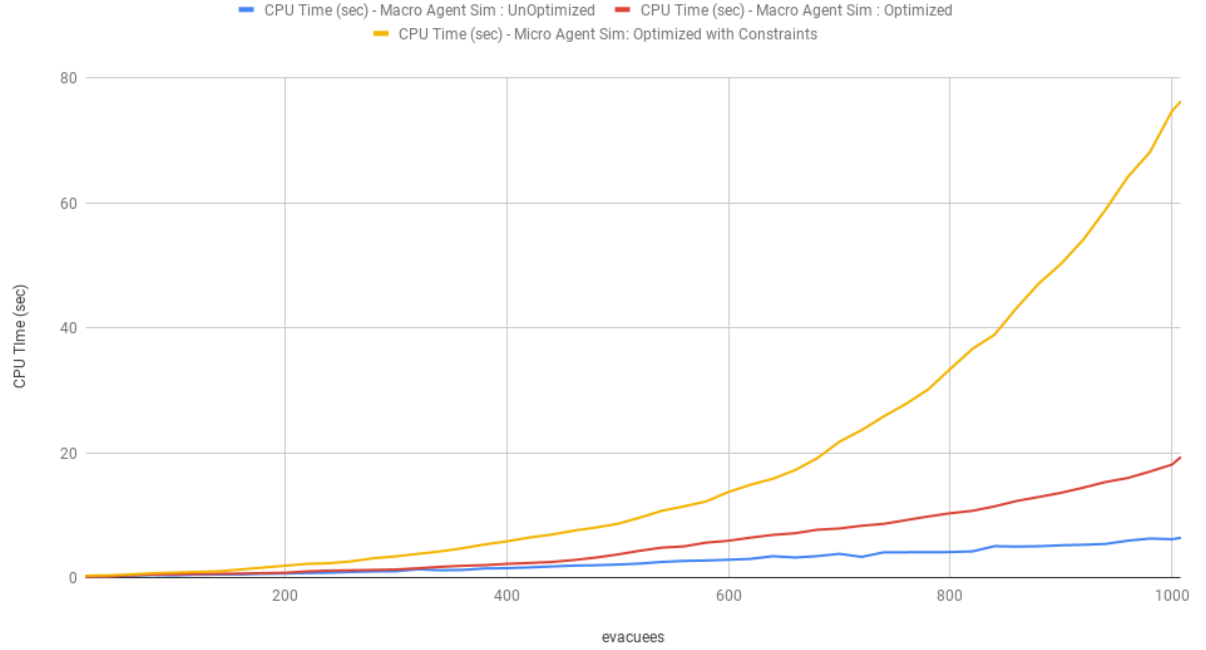
Figure 4.4: Micro vs Macro Agent Simulation

From the figure, the following inferences can be made:

1. The macro agent simulation with no constraints and simulated according to the base algorithm and a constant v_{max} movement velocity has a time complexity of $O(N)$.

2. The macro agent simulation subjected to constraints such as social force definition, area capacity and flow capacity, exhibit a time complexity of $O(N^2)$.

3. Micro agent simulation subjected to varying speeds and grouping in addition to social forces and flow capacity definition for congestion control, exhibit a time complexity of $O(N^3)$.

The implications for these inferences will be discussed in section 4.3.

4.2.3   Case 3: Congestion Dynamics and Flow Control

This section presents a detailed case on the effect of the width of emergency exits on the evacuation time. The work by H. Muccini et al. [24] focuses on an ideal and optimal path without accounting for more advanced constraints. One of the most important effects of a disaster on human agents is the element of chaos and panic. This causes most agents to panic and make rash decisions, all clustering towards an exit point. This causes massive congestions and jams which ultimately leads to stampedes and also the failure for the agents to evacuate to an exist point in the shortest time frame possible. To analyze an optimal egress path however, we do not simulate congestion itself as such, rather we aim for congestion avoidance. To avoid congestion, the routes for various people need to be calculated in real time in order to re-route in the case of an impending congestion scenario.

In order to analyze the congestion, we must first answer the question, what leads to congestion? The answer is two fold as follows:

- The obvious answer is - too little space and too many people to fill that space. The forthcoming tables and graphs are to experiment the effect of exit width on congestion.

- The second answer is more complex, dealing with walking speeds, social forces and group decisions.

In section 4.2.4, detailed case study on walking speeds and decision making is presented. Although the same group dynamics applies to this case too, the focus is on exit width. The forthcoming tables and graph depict the various scenarios that are simulated for both macro and micro agents using the un-optimized and optimized algorithms respectively.

The following table 4.3 depicts optimistic values for macro agents whilst varying exit capacities and thus in turn affecting flow dynamics.

Emergency Exit Width (m)	Evacuation Time Horizon (τ) - Optimistic values for Unoptimized Simulation	Evacuation Time Horizon (τ) - Optimistic Values for Optimized Simulation
0.4	45	83
0.5	43	79
0.6	37	76
0.7	33	73
0.8	28	70
0.9	25	68
1.0	22	62
1.1	20	58
1.2	16	53
1.3	12	48
1.4	11	45
1.5	10	40
1.6	9	37
1.7	9	32
1.8	8	29
1.9	7	25
2.0	6	20
2.1	5	18
2.2	5	15
2.3	5	12
2.4	4	8
2.5	3	6
2.6	3	5
2.7	3	3
2.8	2	2
2.9	2	2
3.0	1	1

Table 4.3: Emergency Exits Width vs Evacuation Time : Optimistic Values for Macro Agent Model

Comparison between the pessimistic cases of optimized and the un-optimized cases are presented for macro agent model below in table 4.4:

Emergency Exit Width (m)	Evacuation Time Horizon (τ) - Pessimistic Values for UnOptimized Simulation	Evacuation Time Horizon (τ) - Pessimistic Values for Optimized Simulation
0.4	120	205
0.5	105	192
0.6	94	181
0.7	86	173
0.8	76	163
0.9	68	155
1.0	62	140
1.1	56	127
1.2	53	111
1.3	48	101
1.4	45	93
1.5	42	80
1.6	37	72
1.7	33	65
1.8	29	52
1.9	25	47
2.0	20	30
2.1	18	19
2.2	15	17
2.3	12	14
2.4	8	11
2.5	6	8
2.6	5	6
2.7	3	5
2.8	2	3
2.9	2	2
3.0	1	1

Table 4.4: Emergency Exits Width vs Evacuation Time : Pessimistic values for Macro Agent Simulation

To run a simulation for the case of micro model agents, certain characteristics were given to 20% of the total number of agents in each case. The characteristics include variable speeds(random speed between 0. 7 m/s and 1.4 m/s), and the other constraints of area capacity and the variable exit door capacity as follows:

Emergency Exit Width (m)	Evacuation Time Horizon (τ) - Optimistic Values for Micro Agent Simulation	Evacuation Time Horizon (τ) - Pessimistic Values for Micro Agent Simulation
0.4	246	415
0.5	223	370
0.6	202	340
0.7	192	306
0.8	184	292
0.9	178	270
1.0	155	240
1.1	140	212
1.2	129	193
1.3	111	163
1.4	101	140
1.5	85	124
1.6	74	112
1.7	65	99
1.8	51	83
1.9	48	60
2.0	31	47
2.1	19	33
2.2	16	28
2.3	13	20
2.4	10	16
2.5	8	9
2.6	7	8
2.7	5	6
2.8	3	3
2.9	2	2
3.0	1	1

Table 4.5: Emergency Exits Width vs Evacuation Time : Micro Agent Simulation

The following series of graphs are a depiction of the above presented tables respectively.
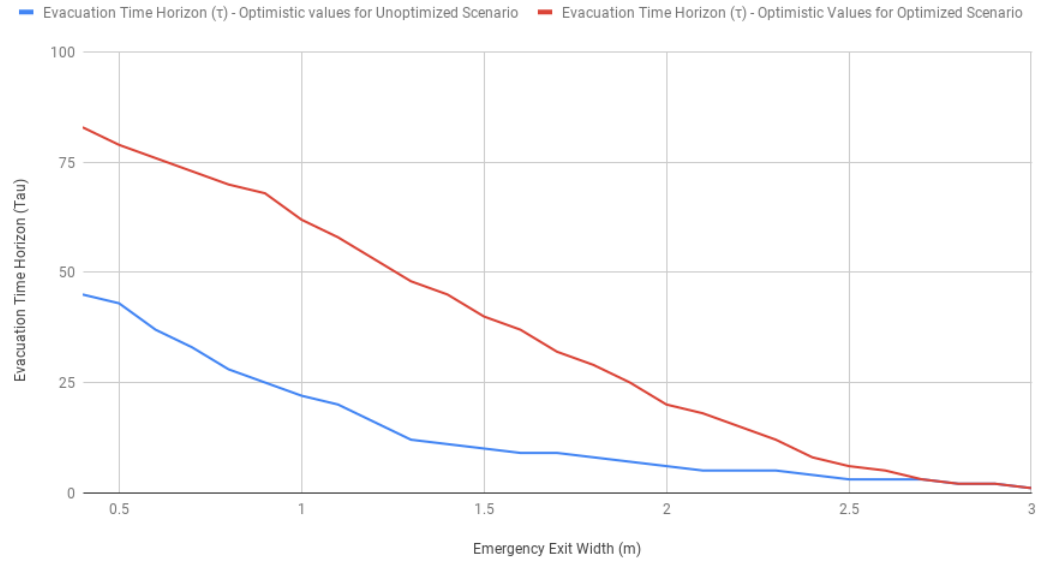

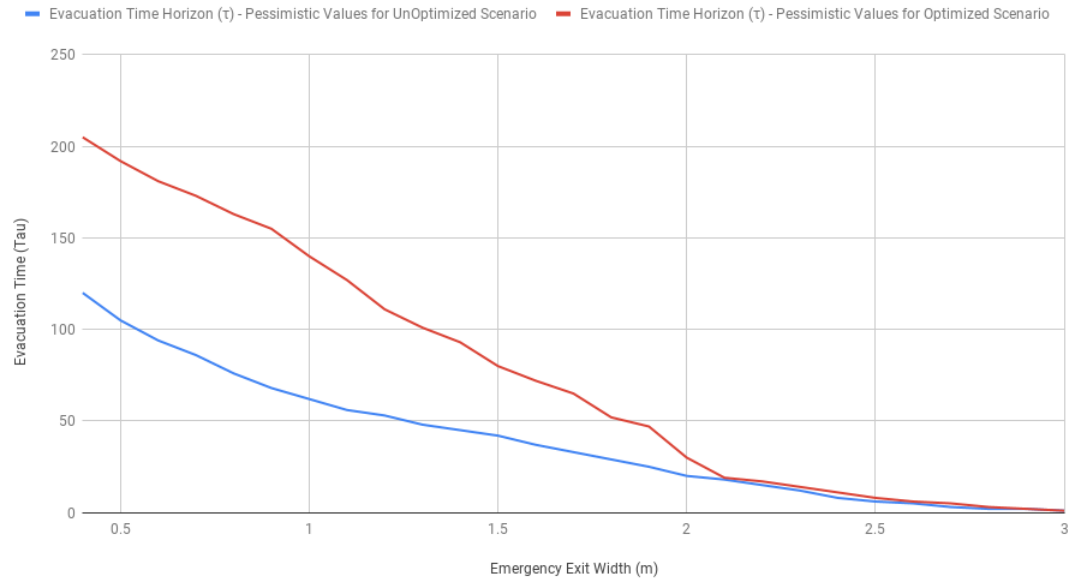Figure 4.5: Optimistic Graph for Macro Agent Simulation

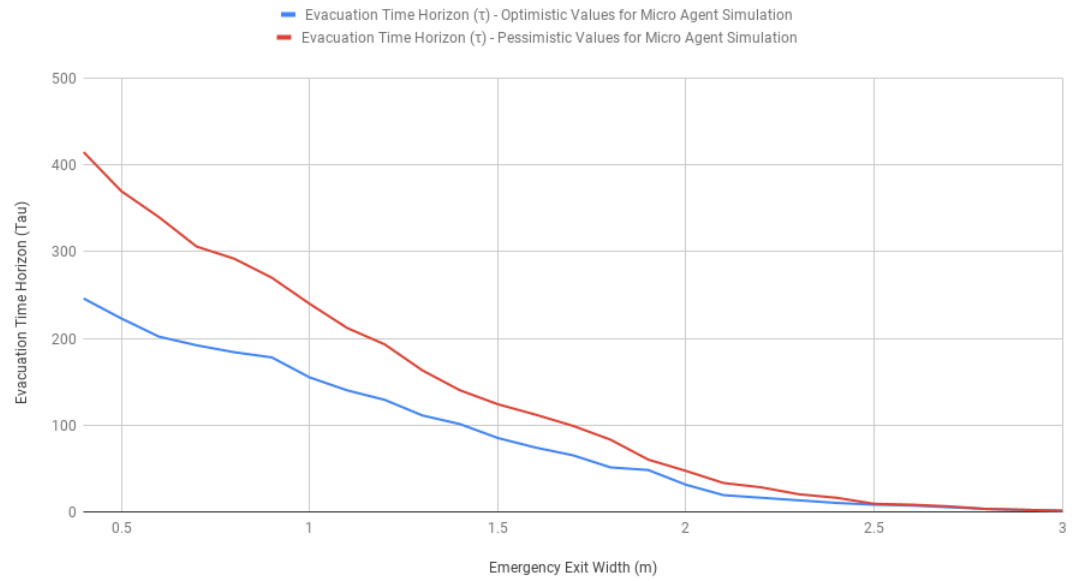Figure 4.6: Pessimistic Graph for Macro Agent Simulation

Figure 4.7: Pessimistic and Optimistic Evacuation Time for Micro Agent Simulation
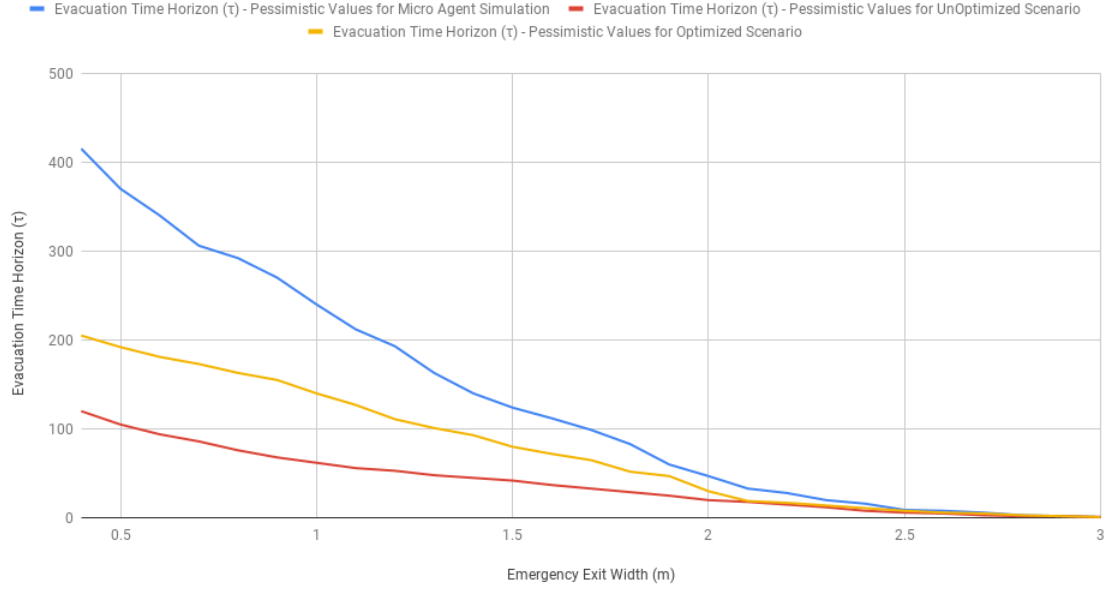
Figure 4.8: Emergency Exits Width vs Evacuation Time : Macro vs Micro Agent Simulation

From the above figures, it is made abundantly clear that the shorter the width of the exits, the more time it would take for the evacuees to evacuate during a disaster scenario. Though this seems like a trivial scenario to depict, it has some serious implications. The most important inference that can be made from the presented data is the derivation of how many people can effectively navigate through to an exit during a disaster scenario. The presented scenario compares the obtained statistical data to the reference figure 8 from [24].

It is also made clear that with more constraints and more realism, the statistical data shows that the congestion tends to increase as the evacuees tend to take longer and longer time to evacuate. Especially in the final scenario with the pessimistic values denoted for the micro model agent simulation, the evacuation time required to get all the agents to safety is exponential compared to the previous cases. The difference between pessimistic case and the optimal one is that in the optimistic case,

only the shortest possible path towards the exit is considered. However, for the pessimistic case, all possible exit strategies are considered and hence the pessimistic case scenario should be considered whilst modeling a safe topology that caters for safe exit paths.

It should also be mentioned that the social distances in the micro agent simulations are set according to the UK fire safety regulations [1] and the correponding guide given at [2]. The guide specifies that the maximum allowed density is 0.3 square meters per person, 0.5 for public houses, 0.8 for exhibition spaces, 1.0 for dining and 2.0 for sports areas etc. From [24], who also follows the same regulations, 1.25 persons per square meter is considered. Hence for this scenario, the social distances constraint is set accordingly. The area capacity constraints that are set throughout all these experiments are set according to these regulation standards. In our case, since its clearly mentioned that not more 6 are allowed within an office space, each room cannot hold more than 6 people at a single time.

Figure 4.8 presents a summary of all the congestion scenarios. The pessimistic values that are obtained from the micro agent scenario shows the difference in the exponentiation graphs where the time taken for evacuation exponentially increases with a time complexity of $O(N^3)$ as the width of the exits are decreased.

4.2.4   Case 4: Shortest Path vs Ideal Path

This section presents the evacuation of agents and the time required based on the scenario of shortest path and the ideal paths. The shortest path in realistic scenario may not always be the shortest time path. Hence it is required to re-route the agents to a more optimal path, i.e. the ideal path in order to reach the exit points. In order to identify the ideal path, the algorithm traveses through all possible paths from the given agent location and then based on flow capacity and congestion ratio,

it strategizes an ideal path for the agents to take whilst heading towards the safe zone.

The ideal path formation depends on some factors which are given below:

- Number of people in the group - this is used in calculation of the ideal path as the number of people in a group can be used to determine if other cluster of agents can form a congestion or not. It falls under congestion control and prediction.

- The movement speed of the group - this is used to calculate the ideal path by predicting if the group can reach a certain location or not to avoid congestion and cluster, if the group can access for instance an emergency service like elevators, emergency stairwells, exits etc.

- Area capacity - This is used to determine if the agent(s) are to be re-routed to another path.

- Door and cell flow dynamic- cell flow dynamic is used to determine if the stream of agents can be navigated through or not, to determine if a certain pathway is one way navigation only or not. The door flow capacity is used to determine the queuing and wait time as most congestions occur at key door and exit points.

The above mentioned conditions are incorporated in this scenario in order to scan through all the possible paths and then determine an ideal path for the agents. Since walking speeds play an important role in determining routing path, it became a necessity to group people according to *Energetic consequences of human sociality: walking speed choices among friendly dyads* by Wagnild et al. [28]. According to Wagnild male agents walk at a faster pace when they are alone at an average speed of 1.53 m/s. They walk the fastest with a male companion at a speed of 1.60 m/s. The speed is subsequently lowered in the presence of a female agent to match her speed,

which comes to around 1.44 m/s. Females walk even slower when accompanied with a female partner at a speed of 1.39 m/s.

The mentioned walking speeds are done so in order to account for the expense of calories spent whilst walking. Men tend to reduce their speeds while walking with a female partner, female agents similarly increase their speeds and the compromized speed averaging at 1.44 m/s is set upon. All the agents presented in the study are capable adults with no apparent health problems.

However in order to make a realistic approach to simulation, all age groups should be considered. Hence for this case, 20% of the total number of agents are considered agents with slower movement speeds, i.e., between 0.7 m/s and 1.2 m/s. All the other agents are given a random speed between 1.2 and 1.6 m/s. The door flow capacity is set to not more than 5 for pessimistic cases as the same value has been used in [24], to remain coherent with the statistics. The upper limit, i.e. optimistic value is no more than 16 persons can pass through a 1 meter wide door during one time slot τ.

In order to model the exact movement speed of an agent within the PedSim environment, the total time required to move an agent across the entire breadth of the topology recorded and then subsequently divided by the total time taken in order to calculate the average speed that is taken by the agent to traverse 1 m of the topology. This then can be used to set variable speeds according to aforementioned constraints and then the scenario is run within the environment of PedSim. The following graphs depict the pessimistic and the optimistic values that are obtained as a result of the simulations. However, for macro agent scenario, a movement speed based on the normal distribution across all agents is set.

The observational data from [24] is used as a baseline reference in order to conduct the forthcoming simulations. For proper analysis and comparison, the initial number of agents present are taken as the same, i.e. $N = 1008$. The study is presented as follows:
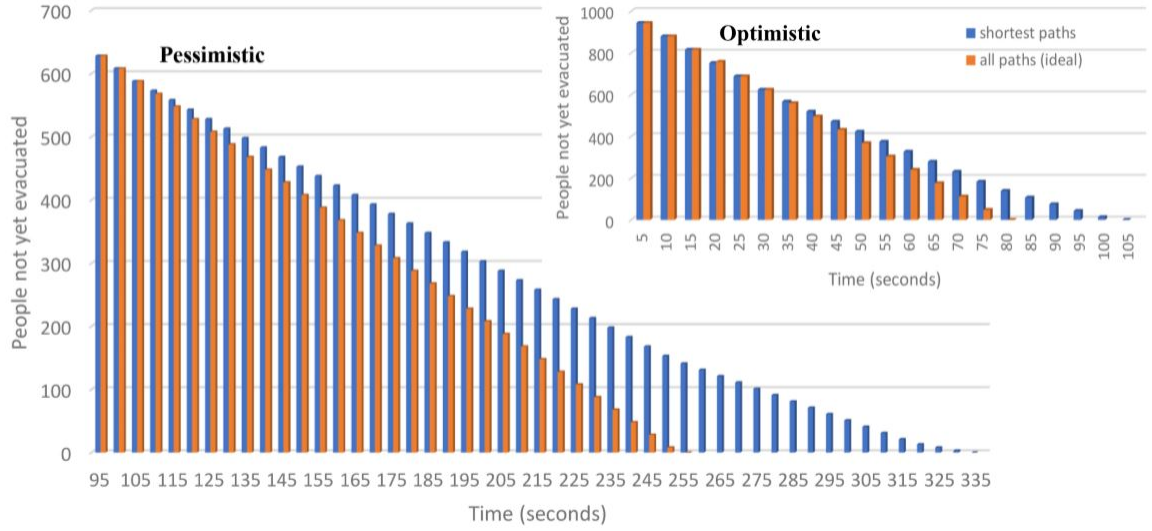
Figure 4.9: Ideal Evacuation vs Shortest Path Evacuation Along Shortest Path for Macro Agents

The above figure represents the number of people who can be evacuated within the given time frame for both pessimistic and optimistic cases, while simulating the whole scene for macro agent scenario. The pessimistic scenario runs for a total of 5 minutes and 35 seconds while the optimistic scenario runs for a total of 1 minute and 20 seconds. The figure below depict the cases for optimistic scenario that run on an macro agent model simulating an optimized scenario for path routing.
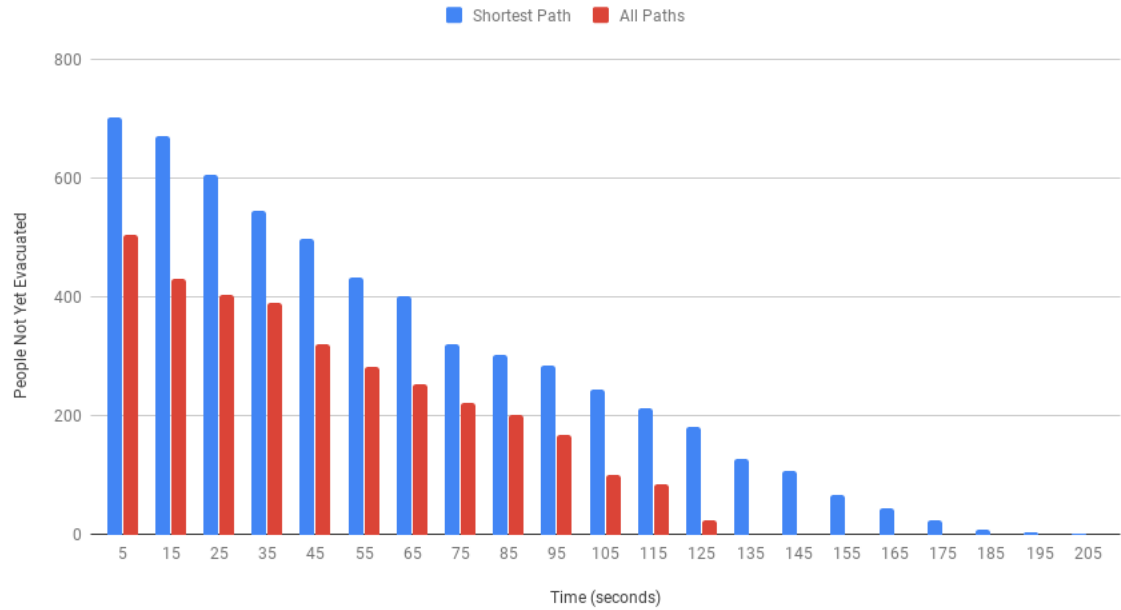
Figure 4.10: Optimistic Graph for Agent Evacuation Using Optimized Macro Agent Simulation

Compared to the previous figure 4.9 depicting the optimistic scenario, its immediately evident that evacuation required is longer compared to and the number of agents that are not evacuated stays on for a longer period of time. This is with an increased capacity for the pathways since its the optimal scenario. The simulation takes a total of 3 minutes and 15 seconds in order to evacuate all people to safety. Some of the inferences made are as follows:

- The interesting result to note is at the end tails of both the graphs. The graph from figure 4.9 shows that the agents that follow the shortest paths tend to be evacuated last compared to the agents that follow the ideal path scenario based on the computation of all path.

- In the optimized macro agent case, we observe a similar trend where in the pattern of behavior as agents that follow either path strategy have similar evacuation times just like figure 4.9.

- This is due to the fact that, in the results provided in 4.9, the agents flow plainly till the third of the total evacuation time and then the agents that follow the shortest path strategy start to experience congestion, while the agents that follow all paths strategy can evacuate quicker.

- A similar phenomenon is experienced in the presented optimized macro agent scenario as depicted in 4.10 as the computation time required for both shortest path and ideal paths are the same for very low number of agents and hence for very low number of agents the evacuation time required is similar. However since the start number of agents are very high, the shortest path strategy experience congestion and hence takes more evacuation time.

- This however becomes very different for very large number of agents and as it can be seen from the graph 4.10, the curve becomes exponential ($O(N^2)$), for the shortest path strategy and increases steadily as the number of agents increase. The increase in number of agents cause higher congestion and hence the exponential increase in the evacuation time.

The next step is to compare the differences in scaling between the pessimistic case for macro agent model simulated in an unoptimized scenario and the pessmisictic case for macro agent model running on the optimized constraints scenario. The prediction is that the rates of scaling will be very drastic as the pessimistic case reduces the flow capacity and also considers a few of the exits blocked. For the purposes of this experiment it was assumed that 2 out of 4 exits were blocked due to a static emergency like earthquake and hence the agents have to be re-routed to safety through the other

exit points. This case is particularly interesting as congestions can be observed and how the agents tackle these congestions through predictive re-routing can be observed.

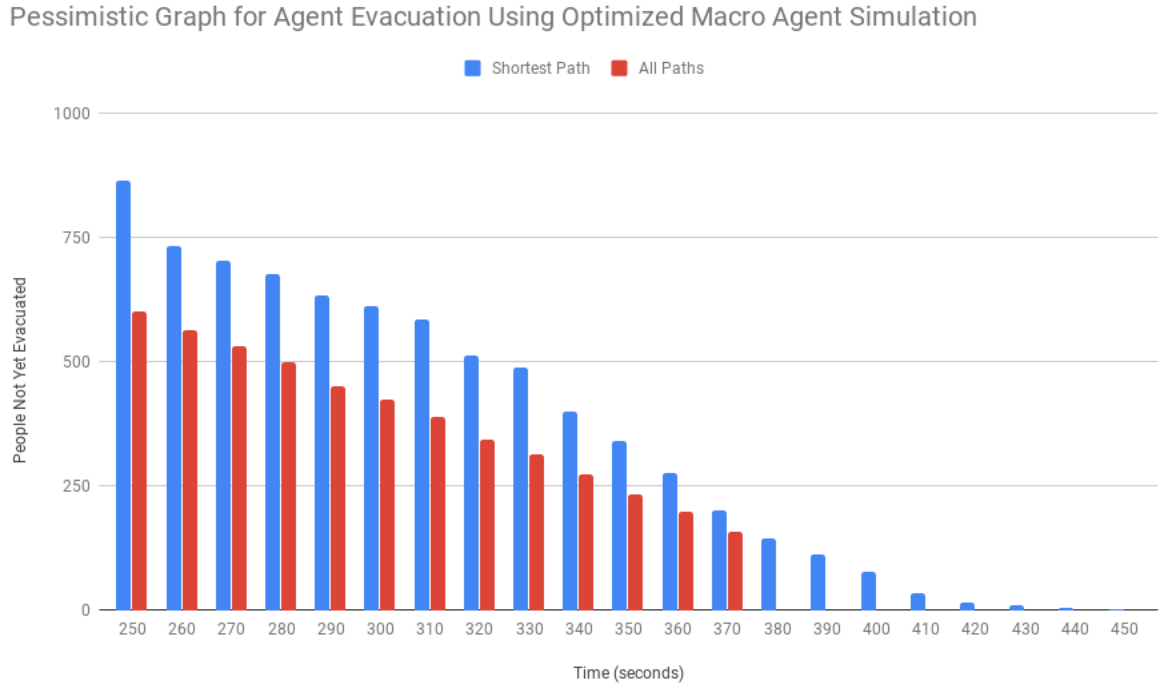The next figure 4.11 presents the aforementioned case.

Figure 4.11: Pessimistic Graph for Agent Evacuation Using Optimized Macro Agent Simulation

The above represented graph typically confirms the previously described hypothesis in terms of scaling. As it can be observed quite clearly, the time required for the evacuation of agents in comparison to the figure 4.9 is almost 1.5 times the time required for the evacuation for the same case presented.

The following are some inferences that are derived from the above graph:

- Evidently the tail end of the graph for agents following the shortest path strategy raises exponentially $(O(N^2))$

- The agents following the all paths strategy only increase linearly as the number of agents increase.

- The implications of these scaling differences are especially important as it can be further broken and analyzed as to which of the constraints cause this peak in scaling. Upon further investigation into the particular constraint causing the computation time difference, it is found that changing the values of area capacity and flow capacity makes most of the difference.

The next two cases present the optimistic and the pessimistic cases of the micro agent model scenario.
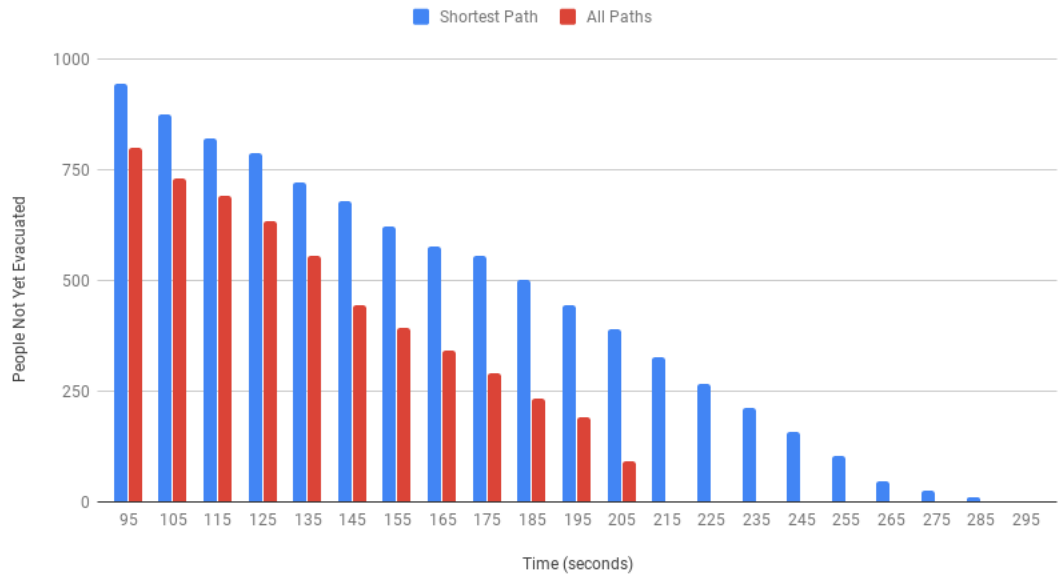

Figure 4.12: Optimistic Case for Ideal Evacuation vs Shortest Path Evacuation Along Shortest Path for Micro Agents

The figure given above depicts an interesting scenario compared to its counter part - the pessimistic case as presented in 4.9. Inferences from the given diagram will be presented below while discussing the following figure.
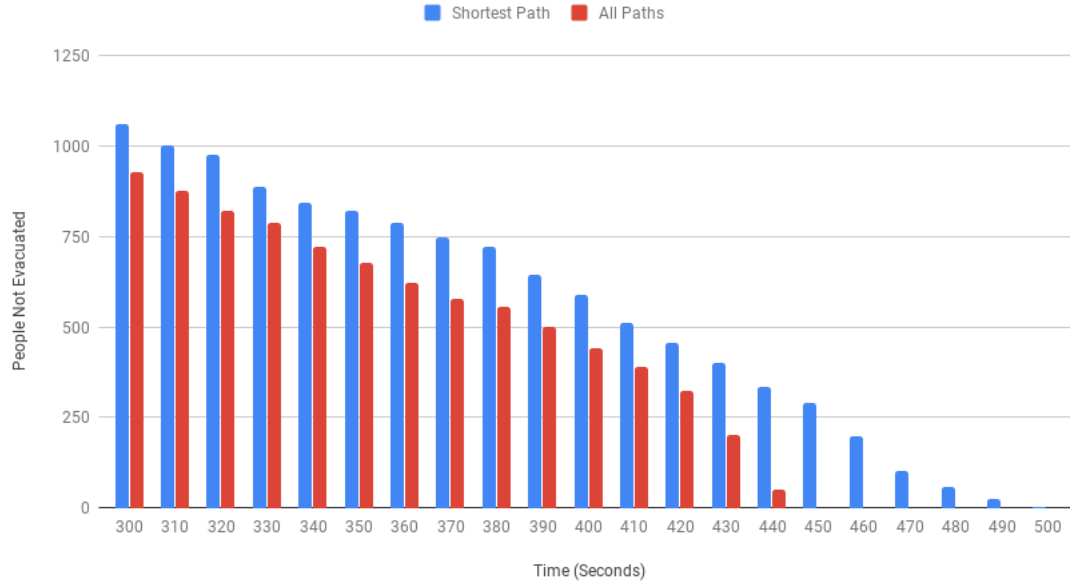
60

Figure 4.13: Pessimistic Case for Ideal Evacuation vs Shortest Path Evacuation Along Shortest Path for Micro Agents

The above two figures depict the interesting differences between the macro and micro agent simulation. As it can be observed from the figures 4.12 and 4.13, it takes longer than the macro agent model simulation in order to evacuate all the people to safety. The following inferences are presented in brief to compare and constrast between the obtained results:

- It is important to note that the simulation performance changes that occur in 4.12 when compared to the results obtained from the optimistic case presented in the figure 4.9.

- In an optimistic scenario, due to higher capacity and a higher degree of resulting congestion avoidance, the all paths strategy functions more optimally compared to its counter.

- This however becomes a more or less even for about 3 quarters of the evacuation time as is shown in figure 4.13. The contrast in the evacuation time occurs due to the following reason:

  - During a pessimistic scenario, it is assumed that 50% of the total emergency exits are inaccessible due to a static emergency blockade. Hence as the numbers dwindle the time differnence between shortest paths and the all paths strategy becomes one and the same. Hence this causes the noticeable difference between the two results given above.

In summary it can drawn to a conclusion that the micro agent simulation with the all paths strategy takes the highest amount of time to evacuate, exhibiting a time complexity of $O(N^3)$. The macro agent model simulated with additional constraints takes a running time of $O(N^2)$.

Chapter 5

Conclusion

From the presented scenario and cases, it is quite evidently evaluated that micro agent simulation, optimized with the applied realistic constraints presents the most realistic approach to evacuation compared to the other cases. Based on this objective presentation, we can design topologies and evacuation systems that are better suited to accommodate the required crowd of pedastrians. Through this thesis, it is also shown how adding even a small constraint to the simulation can severely alter the overall evacuation time of the agents. The nature of these constraints and how they affect the overall evacuation performance of the agents helps us to be better prepared in the likelihood of an actual disaster event.

The present work employs a software architecture that helps improve and optimise the evacuation time which is very crucial in the case of an emergency. From the topology data obtained, it is pipelined into PedSim simulation environment. Once this data and the distribution of people is fed in to the system, the algorithm for evacuation provides a generic initial pathway to the agents and as time progresses, provides an alternate/optimal paths for various agents and suggest the paths suitable to each individual. This is achieved by dividing the floor plan first into blocks and subsequently into number of cells. The cells may have different orientations-horizontal

and vertical. The paths are not allowed to cut through walls. The walls can be in any place even inside the rooms. Safe passage through the doors are then computed which connects to those to safe areas can then be computed. This modular nature of the algorithm makes it easy to scale the versions for future architecture and add more constraints by add-on modules. The simulation tool allows itself to upgraded to more generic pathing algorithm as well. The following are the advantages that are due to the present simulation tool and the employed algorithm:

- All possible trajectories are computed.

- The optimal paths to safe areas are selected from the same.

- Regulates the flow as per the constraints to reduce congestions.

- Erratic and random motions are reduced.

- Evacuation plan is made available to one and all.

- Simulation is carried out in design time, although monitoring the status of evacuation is done in real time.

- Can easily be integrated to IoT based framework.

- Optimal evacuation procedure model can be evolved from the resulting simulations.

5.1   Future Work

This work can be extended to non-standard buildings with additional constraints in future. The internet of things (IoT) helps in locating the number of persons in each block and their location (cell numbers) which can then be dynamically fed in to the simulation tool for optimisation in real time. When such data can be obtained, the simulation can then be shifted from design time to a real time egress path evaluation.

Appendix A

Scenario Definition

This section is used to provide a brief technical description of the implementation details that are involved whilst defining a scenario for the simulation of pedastrian agents within the PedSim environment. The library modules are written entirely in C++ whilst the topology description is described within a .xml file. The simulation tool parses the xml file and graphically projects the topology onto the simulation environment for visual reference. The agent properties are also defined in the xml file such as their base speed, spawn location, total number of agents etc.

The section below provides an example of writing a topology description in the .xml file.

A.1   Defining Topology

The code snippet presented below depicts an example of defining the outline of the building, which was the topology used to run the simulations in. The dimensions of the building are strictly modeled according to the figure 3.1. The dimensions have been scaled up to 5 times in order to suit the PedSim environment. A typical obstacle/wall description is defined as follows:

```
<!-- Outer Walls -->

<!-- TOP SIDE -->
<obstacle x1="-120" y1="-90" x2="-7.5" y2="-90" />
<obstacle x1="-7.5" y1="-90" x2="-7.5" y2="-60" />
<obstacle x1="-7.5" y1="-60" x2="7.5" y2="-60" />
<obstacle x1="7.5" y1="-60" x2="9.5" y2="-60" />
<obstacle x1="17.5" y1="-60" x2="30" y2="-60" />
<obstacle x1="30" y1="-60" x2="30" y2="-90" />
<obstacle x1="30" y1="-90" x2="120" y2="-90" />

<!-- RIGHT SIDE -->
<obstacle x1="120" y1="-90" x2="120" y2="-15" />
<obstacle x1="120" y1="-15" x2="105.5" y2="-15" />
<obstacle x1="97.5" y1="-15" x2="97.5" y2="75" />

<!-- BOTTOM SIDE -->
<obstacle x1="97.5" y1="75" x2="7.5" y2="75" />
<obstacle x1="7.5" y1="75" x2="4.5" y2="75" />
<obstacle x1="-4.5" y1="75" x2="-7.5" y2="75" />
<obstacle x1="-7.5" y1="75" x2="-97.5" y2="75" />
<obstacle x1="-97.5" y1="75" x2="-97.5" y2="90" />
<obstacle x1="-97.5" y1="90" x2="-135" y2="90" />


<!-- LEFT SIDE -->
<obstacle x1="-135" y1="90" x2="-135" y2="-30" />
<obstacle x1="-135" y1="-30" x2="-112.5" y2="-30" />
<obstacle x1="-120" y1="-38" x2="-120" y2="-90" />
```

From the above code snippet, the x and y pair points are given for drawing straight lines. Partial and full agent pathing can be provided for the agents in order for them to start moving towards exit points during an emergency scenario.

```
<!-- waypoints are defined before agents  -->

<waypoint id="w1" x="-75.0" y="-10" r="2" />
<waypoint id="w2" x="-61" y="-3" r="2" />
<waypoint id="w3" x="-50" y="-3" r="2" />
<waypoint id="w4" x="-40" y="-3" r="2" />
<waypoint id="w5" x="-30" y="-3" r="2" />
```

```
<waypoint id="w6" x="-7.5" y="-3" r="2" />
<waypoint id="w7" x="0" y="0" r="2" />
<waypoint id="w8" x="0" y="30" r="2" />
<waypoint id="w9" x="0" y="40" r="2" />
<waypoint id="w10" x="0" y="50" r="2" />
<waypoint id="w11" x="0" y="60" r="2" />
<waypoint id="w12" x="0" y="70" r="2" />
<waypoint id="w13" x="0" y="90" r="2" />
```

The waypoint definition has coordianates x, y and radius r. The .xml file was written by referring the official documentation as provided in [11]. Sample scenarios and tutorials regarding the same are also provided in the same documentation.

The final step in completing the scenario description is to include the agent properties. Given below is a sample agent definition in xml:

```
<agent x="30" y="-30" n="10" dx="50" dy="50">
```

Where,

- n is the number of agents.

- x and y are the spawn location of the agents within the topography of the environment.

- dx and dy are used to determine the spread of the agents spawned. Agents are distributed evenly according to $x - dx$ and $x + dx$ and correspondingly along $y - dy$ and $y + dy$.

Varying n allows for defining the number of agents in a given group. This is particularly useful for the group simulation scenarios. In order to perform micro agent simulations, the n is typically valued as 1. To summarize the agent definition, waypoints for the agents are defined in order to provide a more definitive pathing for the agents based on the spawn locations.

For a micro agent simulation, the agent definition will be as follows:

```
<agent x="30" y="-30" n="1" dx="50" dy="60">
<addwaypoint id="wu" />
<addwaypoint id="wd" />
<addwaypoint id="wz" />
</agent>
<agent x="30" y="-30" n="1" dx="25" dy="30">
<addwaypoint id="wu" />
<addwaypoint id="wd" />
<addwaypoint id="wz" />
</agent>
<agent x="30" y="-30" n="1" dx="94" dy="100">
<addwaypoint id="wu" />
<addwaypoint id="wd" />
<addwaypoint id="wz" />
</agent>
<agent x="30" y="-30" n="1" dx="107" dy="112">
<addwaypoint id="wu" />
<addwaypoint id="wd" />
<addwaypoint id="wz" />
</agent>
```

Once the scenario to be modeled is described in the .xml file as given in the above code snippets, the file is then parsed by the PedSim library modules. Once the data is parsed from the .xml file, the agents are spawned according to the provided definitions and then PedSim proceeds to simulate the agents along the general waypoints as configured from the previous file.

The PedSim library modules are modified and extended in order to simulate the agents according to the algorithm present in [24]. The technical details and description are provided in brief in the next section.

Appendix B

Simulation Implementation

The evacuation of agents within the PedSim environment can be treated as a complex graph problem. Hence the data obtained from the scenario description from the .xml file is translated into a form that can be easily read by the computer as a graph. Since there are many constraints and levels to the topology description, a complex user defined data structure is instantiated to hold this in place.

The PedSim modules have been extended in order to accommodate the topology to graph translation. Two new source files are made and included within the "src" folder. coppito.h and coppito.cpp extends the functionality of the existing PedSim modules in order translate the data. coppito.h defines the prototype for the various functions that are used to instantiate this complex user defined data type.

The implementation is broken down into different modules and are presented as follows. The corresponding prototypes are defined in coppito.h. The first module is to divide the topology into cells. The following code snippet first defines the cell structure and then further expanded as data is populated into the corresponding cell structure.

```cpp
  // Defining a vertex
struct vertex{
 double x;
 double y;
};
 // Defining an edge
struct edge{
 double startx;
 double starty;

 double endx;
 double endy;

 bool wall;
 bool door;
};
// Defining a cell structure
struct cell_structure{
 int cell_number;
 int cell_next[8];
 std::vector<edge> edge_line;
 std::vector<std::vector<edge> > cell_outline;
};
```

The above user defined complex data structure is then used to store data which is given by the following snippet:

```cpp
void Coppito::divide_Cells(){

 //BLOCK 0:
 start.x = -120;
 start.y = -90.0;

 end[0].x = -7.5;
 end[0].y = -90.0;

 end[1].x = -7.5;
 end[1].y = -60.0;

 cell_structure_allocation(start, end, 2);

 //print_cells();
```

```
//BLOCK 1:
start.x = 30.0;
start.y = -90.0;

end[0].x = 120.0;
end[0].y = -90.0;

end[1].x = 120.0;
end[1].y = -60.0;

cell_structure_allocation(start, end, 2);

//BLOCK 2:
start.x = -97.5;
start.y = -45.0;

end[0].x = -7.5;
end[0].y = -45.0;

end[1].x = -7.5;
//end[1].y = 0.0;
end[1].y = 30.0;

cell_structure_allocation(start, end, 2);
}
```

For the ease of implementation, a cluster of cells is taken as a block. The exact number of cells that make up a single block varies. There are 18 blocks in total, as is mentioned previously in section 3.2. The next module after cell division is wall inclusion module. This allows for the algorithm to distinguish cells that have walls and the cells that do not, and also to understand which sides of the four sides of a cell constitutes a wall. The complex data type which contains these list of walls in a cell is defined as the following:

```
std::vector<std::vector<edge> > list_walls;
```

The list_walls is further constitutes the data structure

```
std::vector<vertex> walls;
```

The wall inclusion snippet is further broken down into horizontal and vertical allocation. This is done so because vertical and horizontal walls have different dimensions. Given below is code for horizontal and vertical wall division, and finally invoking the wall inclusion module in order to obtain the list of walls.

```cpp
void Coppito::wall_division_vertical(std::vector<vertex> walls
   ↪ ){
 vertex s; //start wall vertex
 vertex e; //end wall vertex
 vertex t; //temp variable

 s.x = walls[0].x;
 s.y = walls[0].y;

 t = s;

 e.x = walls.back().x;
 e.y = walls.back().y;

 walls.clear();

 //cout << t.x << "," << t.y << endl;
 //cout << e.x << "," << e.y << endl;
 int i = 0;
 while(true){
  if(t.y == e.y)
   break;
  else if(t.y != e.y && t.x == e.x){
   t.x = s.x;
   t.y = s.y;

   //cout << t.x << "," << t.y << endl;
  }

  s.y = s.y + cell_width;
  walls.push_back(t);
  i++;
 }
 edge temp_edge;
 int j = 1;
 //cout << walls.size() << endl;
 for(int i = 0; i < walls.size(); i++){
```

```cpp
    //cout << i << " : " << walls[i].x << "," << walls[i].y <<
        ↪ endl;
    if(walls[j].y != walls.back().y && walls[j].x == walls.back
        ↪ ().x){
     temp_edge.startx = walls[i].x;
     temp_edge.starty = walls[i].y;
     temp_edge.endx = walls[j].x;
     temp_edge.endy = walls[j].y;

     //cout << "(" << temp_edge.startx << "," << temp_edge.
        ↪ starty << ")" << " --> " << "(" << temp_edge.endx <<
        ↪ "," << temp_edge.endy << ")" << endl;

     edge_wall.push_back(temp_edge);
     j++;
    }
    else if(walls[j].x == walls.back().x && walls[j].y == walls.
        ↪ back().y){
     temp_edge.startx = walls[i].x;
     temp_edge.starty = walls[i].y;
     temp_edge.endx = walls.back().x;
     temp_edge.endy = walls.back().y;

     //cout << "(" << temp_edge.startx << "," << temp_edge.
        ↪ starty << ")" << " --> " << "(" << temp_edge.endx <<
        ↪ "," << temp_edge.endy << ")" << endl;

     edge_wall.push_back(temp_edge);
     break;
    }
 }

 list_walls.push_back(edge_wall);
 edge_wall.clear();
}


void Coppito::wall_division_horizontal(std::vector<vertex>
   ↪ walls){
 vertex s; //start wall vertex
 vertex e; //end wall vertex
 vertex t; //temp variable
```

```
s.x = walls[0].x;
s.y = walls[0].y;

t = s;

e.x = walls.back().x;
e.y = walls.back().y;

walls.clear();

//cout << t.x << "," << t.y << endl;
//cout << e.x << "," << e.y << endl;
int i = 0;
while(true){
 if(t.x == e.x)
  break;
 else if(t.x != e.x && t.y == e.y){
  t.x = s.x;
  t.y = s.y;

  //cout << t.x << "," << t.y << endl;
 }

 s.x = s.x + cell_length;
 walls.push_back(t);
 i++;
}
edge temp_edge;
int j = 1;
//cout << walls.size() << endl;
for(int i = 0; i < walls.size(); i++){
 //cout << i << " : " << walls[i].x << "," << walls[i].y <<
    ↪ endl;
 if(walls[j].x != walls.back().x && walls[j].y == walls.back
    ↪ ().y){
  temp_edge.startx = walls[i].x;
  temp_edge.starty = walls[i].y;
  temp_edge.endx = walls[j].x;
  temp_edge.endy = walls[j].y;

  //cout << "(" << temp_edge.startx << "," << temp_edge.
     ↪ starty << ")" << " --> " << "(" << temp_edge.endx <<
     ↪ "," << temp_edge.endy << ")" << endl;
```

```cpp
    edge_wall.push_back(temp_edge);
    j++;
   }
   else if(walls[j].x == walls.back().x && walls[j].y == walls.
      ↪ back().y){
    temp_edge.startx = walls[i].x;
    temp_edge.starty = walls[i].y;
    temp_edge.endx = walls.back().x;
    temp_edge.endy = walls.back().y;

    //cout << "(" << temp_edge.startx << "," << temp_edge.
       ↪ starty << ")" << " --> " << "(" << temp_edge.endx <<
       ↪ "," << temp_edge.endy << ")" << endl;

    edge_wall.push_back(temp_edge);
    break;
   }
  }

  list_walls.push_back(edge_wall);
  edge_wall.clear();
}

void Coppito::wall_allocation(){

  //WALL DIVISION 1:
  wv.x = 30.0;
  wv.y = -90.0;

  walls.push_back(wv);

  wv.x = 120.0;
  wv.y = -90.0;

  walls.push_back(wv);


  wall_division_horizontal(walls);
  walls.clear();

  //WALL DIVISION 2:
  wv.x = 120.0;
```

```
 wv.y = -90.0;

 walls.push_back(wv);

 wv.x =  120.0;
 wv.y = -15.0;

 walls.push_back(wv);


 wall_division_vertical(walls);
 walls.clear();
}
```

The next module is called the door_assignment module and is very similar to the last. This is included since not all walls are doors and any cell that has a side with a door must have a wall too. This makes the traversal of agents from one room to another or passageway to another smooth and efficient.

```
void Coppito::door_assignment(std::vector<vertex> doors){
 edge temp_door;

 temp_door.startx = doors[0].x;
 temp_door.starty = doors[0].y;
 temp_door.endx = doors.back().x;
 temp_door.endy = doors.back().y;

 doorway.push_back(temp_door);
 list_doors.push_back(doorway);
 doorway.clear();
}

void Coppito::door_allocation(){

 //DOOR 0:
 dr.x = -102.5;
 dr.y = -60.0;

 doors.push_back(dr);

 dr.x = -97.5;
 dr.y = -60.0;
```

76

```
doors.push_back(dr);
door_assignment(doors);
doors.clear();
}
```

The final module is the graph_formation module. It is an extremely complex module that tasks in combining all these data into a single extremely complex data structure module.

```
void Coppito::graph_formation(){
 temp_block = block;
 for(int i = 0; i < block.size(); i++){
  for(int j = 0; j < block[i].size(); j++){
   int p = 0;
   //cout << "Cell Number :" << block[i][j].cell_number <<
       ↪ endl;
   for(int k = 0; k < block[i][j].cell_outline.size(); k++){
    for(int l = 0; l < 4; l++){
     for(int a = 0; a < temp_block.size(); a++){
      for(int b = 0; b < temp_block[a].size(); b++){
       for(int c = 0; c < temp_block[a][b].cell_outline.size()
           ↪ ; c++){
        for(int d = 0; d < 4; d++){
         if(block[i][j].cell_outline[k][l].startx ==
             ↪ temp_block[a][b].cell_outline[c][d].endx &&
           block[i][j].cell_outline[k][l].starty == temp_block[
               ↪ a][b].cell_outline[c][d].endy &&
           block[i][j].cell_outline[k][l].endx == temp_block[a
               ↪ ][b].cell_outline[c][d].startx &&
           block[i][j].cell_outline[k][l].endy == temp_block[a
               ↪ ][b].cell_outline[c][d].starty){
          if(temp_block[a][b].cell_outline[c][d].wall != true)
              ↪ {
           if(block[i][j].cell_number != temp_block[a][b].
               ↪ cell_number){
            block[i][j].cell_next[p] = temp_block[a][b].
                ↪ cell_number;
            p++;
           }
          }
```

```
                    else if(temp_block[a][b].cell_outline[c][d].wall ==
                       ↪ true){
                    if(temp_block[a][b].cell_outline[c][d].door == true
                       ↪ ){
                    if(block[i][j].cell_number != temp_block[a][b].
                       ↪ cell_number){
                    block[i][j].cell_next[p] = temp_block[a][b].
                       ↪ cell_number;
                    p++;
                    }
                   }
                   else
                    continue;
                 }
                }
                else if(block[i][j].cell_outline[k][l].startx ==
                   ↪ temp_block[a][b].cell_outline[c][d].startx &&
                 block[i][j].cell_outline[k][l].starty == temp_block[
                    ↪ a][b].cell_outline[c][d].starty &&
                 block[i][j].cell_outline[k][l].endx == temp_block[a
                    ↪ ][b].cell_outline[c][d].endx &&
                 block[i][j].cell_outline[k][l].endy == temp_block[a
                    ↪ ][b].cell_outline[c][d].endy){
                 if(temp_block[a][b].cell_outline[c][d].wall != true)
                    ↪ {
                  if(block[i][j].cell_number != temp_block[a][b].
                     ↪ cell_number){
                   block[i][j].cell_next[p] = temp_block[a][b].
                      ↪ cell_number;
                   p++;
                  }
                 }
                 else if(temp_block[a][b].cell_outline[c][d].wall ==
                    ↪ true){
                  if(temp_block[a][b].cell_outline[c][d].door == true
                     ↪ ){
                   if(block[i][j].cell_number != temp_block[a][b].
                      ↪ cell_number){
                   block[i][j].cell_next[p] = temp_block[a][b].
                      ↪ cell_number;
                   p++;
                   }
                  }
```

```
              else
               continue;
}}}}}}}}}}}
```

These core modules make up the scenario for the simulation. Once the simulation scenario is defined as above, the simulation is then set to simulate with the additional constraints by tweaking the values as presented from the figure 3.4. Additional tutorial and documentation with regards to PedSim and its library modules are available in official site `http://pedsim.silmaril.org/`

The full implementation of the above code is made publicly available in the following github link :

`https://github.com/hrishikesh9409/Pedsim_Thesis/tree/master/coppito0`.

Bibliography

[1] Fire safety in the workplace. `https://www.gov.uk/workplace-fire-safety-your-responsibilities`, 2019. [Online; accessed 15-03-2019].

[2] Merseyside Fire and Rescue Service. `https://www.merseyfire.gov.uk/aspx/pages/protection/pdf/Calculating_Occupancy_assembly_buildings_GT.pdf`, 2019. [Online; accessed 15-03-2019].

[3] Sameera Abar, Georgios K Theodoropoulos, Pierre Lemarinier, and Gregory MP OHare. Agent based modelling and simulation tools: a review of the state-of-art software. *Computer Science Review*, 24:13–33, 2017.

[4] Prasad Annadata, Wisam Eltarjaman, and Ramakrishna Thurimella. Person detection techniques for an iot based emergency evacuation assistance system. In *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*, MOBIQUITOUS 2016, pages 77–82, New York, NY, USA, 2016. ACM.

[5] William H Avery and Jamie Soo. *Emergency/disaster guidelines and procedures for employees*. CCH Canadian Limited, 2003.

[6] Georgia Bateman and Arnab Majumdar. An assessment of the simulation modeling of emergency evacuations in sports stadiums. 01 2018.

[7] Neeti Bhargava, V Katiyar, Mukat Sharma, and P Pradhan. Earthquake prediction through animal behavior: A review. *Indian Journal of Biomechanics: Special Issue*, 78, 01 2009.

[8] Po-Han Chen and Feng Feng. A fast flow control algorithm for real-time emergency evacuation in large indoor areas. *Fire Safety Journal*, 44(5):732–740, 2009.

[9] Jehyun Cho, Ghang Lee, Jongsung Won, and Eunseo Ryu. Application of dijkstra's algorithm in the smart exit sign. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 31, page 1. IAARC Publications, 2014.

[10] Wonjoon Choi, Horst W Hamacher, and Suleyman Tufekci. Modeling of building evacuation problems by network flows with side constraints. *European Journal of Operational Research*, 35(1):98–110, 1988.

[11] Gloor Christian. Pedastrian Crowd Simulator. `http://pedsim.silmaril.org/documentation/libpedsim/libpedsim.pdf`, 2019. [Online; accessed 15-03-2019].

[12] Damon P. Coppola. Chapter 1 - the management of disasters. In Damon P. Coppola, editor, *Introduction to International Disaster Management (Third Edition)*, pages 1 – 39. Butterworth-Heinemann, Boston, third edition edition, 2015.

[13] Damon P. Coppola. Chapter 4 - mitigation. In Damon P. Coppola, editor, *Introduction to International Disaster Management (Third Edition)*, pages 224 – 274. Butterworth-Heinemann, Boston, third edition edition, 2015.

[14] W Daamen and SP Hoogendoorn. Emergency door capacity: influence of door width, population composition and stress level. *Fire technology*, 48(1):55–71, 2012.

[15] Antoine Desmet, Huibo Bi, and Erol Gelenbe. Simulation of congestion management for emergency evacuation.

[16] John El Khoury, Jean-Paul M Arnaout, and Caline El Khoury. Evacuation of a highly congested urban city: the case of beirut. 2015.

[17] E. Gelenbe, , and E. Seref. Cognitive packet networks. In *Proceedings 11th International Conference on Tools with Artificial Intelligence*, pages 47–54, Nov 1999.

[18] Doungmo Goufoc, Abiola P Chokkid, and Maurice V Yengoa. Agent-based modelling and simulation for evacuation of people from a building in case of fire. *Procedia Computer Science*, 130:10–17, 2018.

[19] Steve Gwynne, Edward R Galea, M Owen, Peter J Lawrence, and L Filippidis. A review of the methodologies used in the computer simulation of evacuation from the built environment. *Building and environment*, 34(6):741–749, 1999.

[20] Shinkishi Hatai and Noboru Abe. The responses of the catfish, parasilurus asotus, to earthquakes. *Proceedings of the Imperial Academy*, 8(8):375–378, 1932.

[21] Motoji Ikeya, Hiroshi Furuta, Naemi Kajiwara, and Hiroko Anzai. Ground electric field effects on rats and sparrows: Seismic anomalous animal behaviors (saabs). *Japanese journal of applied physics*, 35(8R):4587, 1996.

[22] Margrethe Kobes, Ira Helsloot, Bauke De Vries, Jos G Post, Nancy Oberijé, and Karin Groenewegen. Way finding during fire evacuation; an analysis of unannounced fire drills in a hotel at night. *Building and Environment*, 45(3):537–548, 2010.

[23] Marin Lujak, Holger Billhardt, Jürgen Dunkel, Alberto Fernández, Ramón Hermoso, and Sascha Ossowski. A distributed architecture for real-time evacuation guidance in large smart buildings. 2017.

[24] Henry Muccini, Claudio Arbib, Paul Davidsson, and Mahyar Tourchi Moghaddam. An iot software architecture for an evacuable building architecture. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[25] Francesco Palmieri, Massimo Ficco, Silvio Pardi, and Aniello Castiglione. A cloud-based architecture for emergency management and first responders localization in smart city environments. *Computers & Electrical Engineering*, 56:810–830, 2016.

[26] Eleanor M. Simonsick, Luigi Ferrucci, Magdalena I. Tolea, Antonio Terracciano, Paul T. Costa, Michael Griswold, Samer S. Najjar, Angelo Scuteri, Barbara Deiana, Marco Orr, Manuela Uda, Marco Masala, and David Schlessinger. Sex-Specific Correlates of Walking Speed in a Wide Age-Ranged Population. *The Journals of Gerontology: Series B*, 65B(2):174–184, 01 2010.

[27] Ashutosh Trivedi and Shrisha Rao. Agent-based modeling of emergency evacuations considering human panic behavior. *IEEE Transactions on Computational Social Systems*, 5:277–, 03 2018.

[28] Janelle Wagnild and Cara M Wall-Scheffler. Energetic consequences of human sociality: walking speed choices among friendly dyads. *PloS one*, 8(10):e76576, 2013.

[29] Jianhong Ye, Xiaohong Chen, Chao Yang, and Jiaorong Wu. Walking behavior and pedestrian flow characteristics for different types of walking facilities. *Transportation Research Record: Journal of the Transportation Research Board*, (2048):43–51, 2008.