ENHANCED COMPUTATIONAL MODELING OF URBAN EVACUATION: A REVERSE DYNAMIC DIJKSTRA-BASED EXIT ALLOCATION AND FIRE-SPREAD SIMULATION IN MESOSCOPIC CELLULAR AUTOMATA

SUBMITTED TO THE FACULTY OF THE
DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF INFORMATION TECHNOLOGY AND COMPUTING
BY

SEAN VERGEL LABNOTIN
REGGIE YAP. HERMOSISIMA
KENNETH LUZA. LUMOD
JOHN VINCENT ABCEDE. PARBA

IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE
DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

DECEMBER 5, 2025

Chapter I

INTRODUCTION

1 Introduction

Fires in densely populated urban areas pose catastrophic risks, making effective evacuation strategies essential to minimizing casualties and damage (Patac & Vicente, 2019; World Health Organization, 2023). While Cellular Automata (CA) models, particularly the balanced Mesoscopic Cellular Automata (MCA), are valuable tools for simulation, current evacuation strategies based on static, proximity-based exit allocation are insufficient. These models often lead to predictable congestion and stampedes at dominant exits (Tordeux et al., 2018; Oyola et al., 2017). To address this, we propose enhancing the MCA framework by implementing a Reverse Dynamic Dijkstra algorithm. This method adapts exit allocation in real-time, optimizing routes based on exit distance, congestion, and penalty fields to simulate more realistic evacuee behavior. The goal is to improve evacuation safety and throughput compared to traditional MCA models.

1.1 Background of the study

Urban fires are still one of the most devastating dangers, which affect modern communities, particularly in densely populated areas. The issue of fire-related disasters is getting bigger and less manageable as the area of cities and their impact grow accordingly. The urban population across the world has doubled within the last 70 years global urban population has increased by 25% to 50%—with the quickest growth happening in low-income countries

(Stevens & Rush, 2025). This sudden urbanization raises the fire risk to a higher level, especially in underprivileged and overcrowded areas where just one ignition can cause massive destruction (Patac & Vicente, 2019).

Globally, fires are responsible for approximately 180,000 deaths each year, with the burden disproportionately concentrated in low-income countries. Regions such as Africa and South-East Asia account for nearly two-thirds of all fire-related fatalities (World Health Organization, 2023). The Philippines frequently experiences severe urban fires, exemplified by the 2017 Manila incident that displaced more than 15,000 residents (Villamor & Goldman, 2017). These events highlight the urgent need for effective evacuation planning, especially in large institutional and residential environments.

In order to facilitate such planning, computational models have turned into indispensable resources for the examination of fire dynamics and the forecasting of crowd behavior in emergencies. Cellular Automata (CA)-driven models have emerged as the most important ones due to their potential of imitating the intricate fire spreading and pedestrian interactions through simple, local rules. However, every CA scale has its own drawback: on one hand, the microscopic CA models are very detailed but in large areas, they are limited by high computational costs, whereas the macroscopic ones are the other extreme because they are economical in terms of computation but are

likely to oversimplify walking people's behavior (Tordeux et al., 2018; Li & Xu, 2021).

Despite these advantages, traditional MCA evacuation models typically assign evacuees to the nearest exit based solely on distance. This can produce severe congestion at specific exits while leaving others underutilized, increasing evacuation time and potentially elevating risk in dense fire scenarios. To mitigate this, pathfinding algorithms such as Dijkstra's have been incorporated to compute optimal routes beyond simple proximity (Oyola et al., 2017). However, recalculating shortest paths from every source in large networks remains computationally demanding.

Building on these developments, this study enhances the MCA model of Lv et al. (2021) by integrating a reverse-routing approach with dynamic congestion awareness. This enables adaptive exit allocation that considers not only geometric distance but also real-time hazard conditions and pedestrian density. The result is a more robust and realistic evacuation framework aimed at improving safety and reducing overall evacuation time during urban fire scenarios.

1.2 Problem Statement

The purpose of mesoscopic Cellular Automata (CA) models as decision-support tools for fire evacuation planning is greatly limited by two principal drawbacks: the difficulty of inefficient exit allocation and the

absence of full hazard integration. Evacuation in current models are mostly based on the closeness of the refugees to the exits, which is a primitive method pointed out by Xu et al. (2020) and Lv et al. (2021) to result in the formation of crowds and jams at the main exits causing long delays in the evacuation process while other exits remain almost unused. Besides that, a vast majority of the simulations based on Cellular Automata do not take into account the dynamic hazards like fire, debris, or smoke to the full extent; something that Liu et al. (2023) have indicated leads to the creation of impractical routes that do not respond to the changing danger. This is a serious drawback since the hazards of the moment, particularly smoke, greatly diminish the visibility, walking speed, and route availability, thus making the whole exercise unrealistic and of little use in campus and built-environment scenarios.

1.3 Objectives of the Study

This research aims to develop an enhanced Mesoscopic Cellular Automata (MCA) model—building upon the framework established by Lv et al. (2021)—specifically to minimize casualties. This is achieved by incorporating a Dijkstra routing algorithm for optimized movement and additional penalties to introduce greater realism into the evacuation simulation.

1.3.1. Improve exit allocation and reduce bottlenecks

Develop and implement an adaptive exit-allocation mechanism within the MCA framework by replacing the dynamic floor field with a

Reverse Dynamic Dijkstra routing field enhanced by a congestion-penalty term. Unlike the standard Dijkstra algorithm, which is optimized for a single source, the reverse formulation treats all exits as simultaneous sources, making it more computationally efficient and suitable for multi-exit evacuation scenarios (Zhu & Sun, 2021).

1.3.2. Integrate hazard-aware routing using probabilistic fire coupling

Extend the enhanced MCA model by embedding hazard-related penalties—such as fire, smoke, and debris—into the congestion penalty field, in addition to overcrowding effects. Following the probabilistic principles of fire spread modelling described by Parisien et al. (2019), this objective examines how evolving hazards influence total evacuation time and overall system performance.

1.4 Scope and Limitations of the Study

This section outlines the specific boundaries of the research—what the study covers, the modelling features included, and the conditions under which the evacuation simulations operate—as well as the inherent constraints and assumptions that may affect the generalizability and precision of the results.

1.4.1 Scope

This study focuses on simulating fire evacuation scenarios within university campuses, using maps derived from publicly available campus layouts. The model incorporates the fire-spread evacuation framework proposed by Lv, Wang, Fang, and Mao (2021), and enhances it through a Reverse Dynamic Dijkstra routing mechanism combined with congestion-penalty fields.

Unlike traditional implementations—where individuals' routes are computed using the standard forward Dijkstra algorithm (Mirahadi & McCabe, 2020; Rahayuda & Santiari, 2021)—the proposed Reverse Dijkstra approach computes evacuation paths from exits outward. This formulation reduces redundant path computations across multiple population sources, improves computational efficiency, and enables dynamic re-routing as environmental conditions change.

The enhanced pathfinding mechanism also supports multi-path allocation and will be tested within the Mesoscopic Cellular Automata (MCA) framework for urban evacuation defined by Lv et al. (2021). Model performance will be evaluated using eight standardized MCA performance metrics: total evacuation time, peak evacuation flow rate, overall and per-exit flow rates, the number of remaining evacuees over time, exit-wise evacuee counts, evacuee density in observed cells, and spatial distribution of evacuees.

1.4.2 Limitations

The study acknowledges several limitations.

(1) Human behavior is simplified within the simulation. The model relies primarily on route planning and does not explicitly incorporate psychological dynamics, panic effects, or the behavior of injured individuals (Zhou, Cai, & Zhou, 2019).

(2) Fire-spread dynamics are modeled using probabilistic rules rather than detailed physical fire-spread algorithms. Although this approach improves computational efficiency, it introduces a degree of unrealism and may reduce the precision of fire dynamics representation (Parisien, Dawe, Miller, Stockdale, & Armitage, 2019).

(3) The campus models are limited to ten selected universities. Road networks are represented as segmented road cells rather than fine-grained grids, a simplification that reduces computational cost while still maintaining an accurate depiction of pedestrian density and flow (Lv, Wang, Fang, & Mao, 2021).

(4) The results are specific to the selected university environments and may not be directly generalizable to other geographic or urban contexts without model adjustments.

(5) Finally, computational constraints such as grid resolution, processing capacity, and available runtime limit the overall scale and fidelity of simulation scenarios.

1.4 Significance of the Study

This study offers significant value by advancing Mesoscopic Cellular Automata (MCA) evacuation modeling. Academically, it bridges graph-based pathfinding (Dijkstra) with cellular automata dynamics, a novel integration. It specifically extends the static Floor-Field approach (Lv et al., 2021) by implementing a Reverse Dynamic Dijkstra mechanism and a congestion-penalty formulation. Drawing theoretical support from existing exit-rooted distance propagation methods (Li et al., 2019), these enhancements directly mitigate known limitations of floor-field models, such as overcrowding and herding effects.

The study yields a scalable, adaptive evacuation framework applicable to real-world environments. This framework leverages the proven computational efficiency of reverse labeling dynamic Dijkstra search for multi-exit networks (Oyola, Romero, & Vintimilla, 2017) and integrates congestion-aware modeling, which is effective in mitigating bottlenecks and reducing stampede risks during emergencies (Mirahadi & McCabe, 2020).

For university campuses, the proposed Reverse Dynamic Dijkstra-based MCA model can support disaster-preparedness planning and real-time evacuation management. More broadly, it provides a transferable framework for other densely populated facilities, offering practical insights to minimize congestion, improve evacuation efficiency, and ultimately reduce potential casualties in urban fire scenarios.

Chapter II

REVIEW OF RELATED LITERATURE

2.1 Evacuation Model Classifications

Evacuation models are commonly grouped into microscopic, macroscopic, and mesoscopic frameworks, each suited to different spatial and computational requirements. Microscopic models simulate the movement and interaction of individual agents and can reproduce complex behaviours such as lane formation and competitive conflicts. However, their computational cost increases sharply with population size, making them unsuitable for campus-scale scenarios (Helbing, Farkas, & Vicsek, 2000).

Macroscopic models treat crowds as continuous flows, providing computational efficiency but lacking the ability to represent local heterogeneity or route-level behaviors. Mesoscopic models bridge these extremes by grouping evacuees into aggregated cells or packets, preserving spatial resolution while remaining computationally efficient for large and complex environments such as campuses or districts (Tordeux et al., 2018; Shi, Lee, & Ma, 2018). These characteristics make Mesoscopic Cellular Automata (MCA) particularly appropriate for large-area fire evacuation studies.

2.2 Fire-Spread and Data-Driven CA Models

Cellular Automata are widely used to simulate fire behaviour due to their ability to capture spatial contagion and temporal dynamics. Patac and

Vicente (2019) introduced a data-driven CA fire model that incorporates ignition susceptibility learned via Extreme Learning Machines (ELM). Their results show that CA can represent heterogeneous fire spread patterns while remaining computationally lightweight. Such hazard maps are valuable layers for evacuation simulations, influencing route viability and risk-aware pedestrian movement.

2.3 CA Approaches to Pedestrian Dynamics

CA-based pedestrian models—particularly floor-field models—capture emergent behaviours such as clogging, arching, faster-is-slower effects, and directional lane formation (Burstedde et al., 2001). Static fields encode attraction toward exits, while dynamic fields reflect local interactions and congestion. Extensions including friction, conflict resolution, and repulsion mechanisms provide higher realism under evacuation conditions (Burstedde et al., 2001; Helbing et al., 2000).

Because CA models are modular, additional influences such as visibility reduction from smoke, fire proximity, terrain difficulty, or temporary obstacles can be seamlessly integrated into the movement rules.

2.4 Grid-Based Mesoscopic CA Models

Grid-based mesoscopic CA represents campus-scale environments using coarse cells that track aggregated occupant counts and flow. Shi et al.

(2018) demonstrated that such models significantly reduce computation time while retaining enough spatial detail to capture congestion and bottleneck formation. Tordeux et al. (2018) proposed hexagonal mesoscopic grids that improve movement directionality and reduce discretization artifacts.

More recent works—such as Lv, Wang, Fang, and Mao (2023)—use road-segment mesoscopic partitioning that aligns directly with walkable networks, allowing efficient simulation across large outdoor environments. Qiu et al. (2024) emphasized the importance of time-step granularity, showing that coarse steps distort congestion evolution and evacuation time, motivating careful sensitivity testing.

2.5 Fire and Smoke Dynamics in Evacuation Models

Fire and smoke critically influence evacuation due to decreased visibility, heat exposure, and blocked passages. Liu, Li, and Sun (2023) integrated Fire Dynamics Simulator (FDS) outputs—visibility and CO concentration—into a CA pedestrian model, demonstrating significant delays and rerouting when hazard layers were included.

Other studies explored coupled fire–evacuation systems where pedestrian congestion alters smoke accumulation or fire growth (Li et al., 2023/2024). These models reinforce that evacuation frameworks must dynamically incorporate hazard fields rather than relying solely on static routing.

2.6 Graph-Based Routing Approaches

Shortest-path algorithms are widely used for evacuation routing. Standard Dijkstra efficiently computes optimal paths but assumes static, time-invariant edge costs, which becomes unrealistic when hazards or congestion evolve. To address this, studies have proposed hazard-aware cost functions and dynamic rerouting strategies that incorporate fire intensity, visibility reduction, or walkability penalties (Oyola, Romero, & Vintimilla, 2017; Samah, 2015).

Mirahadi and McCabe's EvacuSafe (2020) exemplifies risk-based routing, integrating hazard exposure, exit conditions, and congestion. Capacity-aware extensions such as CASPER and congestion-penalized Dijkstra variants show that redistributing evacuee loads reduces bottlenecks and improves evacuation times (Shahabi et al., 2014).

2.7 Congestion Prediction and Capacity-Aware Routing

Congestion plays a critical role in evacuation efficiency. Dynamic frameworks like EvacuSafe quantify route risk by incorporating density, fire risk, and visibility (Mirahadi & McCabe, 2020). Robotics and evacuation research consistently show that penalizing high-density edges yields smoother flow and balanced exit usage (Shahabi et al., 2014). Yue et al. (2025) demonstrated that failing to consider interactions between fire progression and congestion leads to systematic underestimation of evacuation times.

In mesoscopic CA models, density penalties can be directly embedded in cell transition rules, enabling real-time congestion-sensitive routing without major computational overhead.

2.8 Hybrid MCA–Routing Approaches

Several studies highlight the limitations of MCA models that rely solely on proximity-based exit choice, which results in herding and severe congestion patterns. Hybrid models combine MCA with global routing mechanisms typically Reverse Dijkstra fields to overcome this limitation.

Reverse-search or multi-source Dijkstra methods compute a global distance-to-exit field and allow directional guidance at every cell. When updated dynamically, these fields adjust to blocked or hazardous routes (Shi et al., 2018; Lv et al., 2021; Mirahadi & McCabe, 2020). Such hybridization improves route allocation, distributes evacuees more evenly across exits, and offers scalability for large outdoor networks.

2.9 Synthesis of Research Gap

Existing mesoscopic CA evacuation models offer strong scalability for large environments, but they still suffer from two major unresolved limitations. First, most mesoscopic CA studies continue to rely on proximity-based exit assignment, which causes herding and severe bottlenecks when multiple exits exist. Lv et al. (2021) and Shi et al. (2018)

show that mesoscopic models can capture flow dynamics, but they still allocate exits based on geometric distance rather than global route optimality. Although global routing methods such as Reverse Dijkstra fields have been proposed in broader evacuation routing research (Oyola, Romero, & Vintimilla, 2017), these approaches have not been fully integrated into mesoscopic CA for campus-scale evacuation.

Second, prior research demonstrates limited coupling between dynamic hazards and routing decisions. Fire-spread CA models (Patac & Vicente, 2019), smoke-aware evacuation systems (Liu, Li, & Sun, 2023), and congestion-penalized routing studies (Mirahadi & McCabe, 2020; Yue et al., 2025) show that hazards significantly alter route safety and movement capacity. However, most mesoscopic CA implementations treat hazards as passive modifiers of speed or visibility rather than as factors that actively reshape global route selection. Consequently, evacuees may continue following routes that were optimal initially but have since become unsafe due to spreading fire, increasing smoke density, debris, or congestion.

These gaps highlight a critical need for an evacuation approach that (1) replaces proximity-driven exit assignment with a globally computed routing field, and (2) dynamically updates route costs in response to evolving hazard and congestion conditions. The present study addresses these limitations by combining a Reverse Dynamic Dijkstra routing field (Oyola et al., 2017) with explicit hazard-penalty formulations derived from fire and pedestrian-dynamics literature (Liu et al., 2023; Yue et al., 2025). This enables

a scalable, hazard-aware mesoscopic CA that more accurately reflects realistic evacuation behaviour under dynamic fire conditions.

Chapter III

METHODOLOGY

3.1 Methodology Overview

This study is based on the Mesoscopic Cellular Automata (MCA) framework of Lv et al. (2021), which serves as the baseline model for simulating pedestrian flow and exit selection. The proposed method enhances this framework by adding congestion and hazard penalties, a Reverse Dynamic Dijkstra routing mechanism, and safe-zone rerouting to produce more adaptive evacuation behavior. The enhanced model follows the same mesoscopic cellular automaton of Lv et al. (2021) structure but improves its routing and hazard responsiveness. All simulations are performed on mesoscopic campus road-cells, and results are evaluated using standardized evacuation metrics.

3.2 Baseline Framework and Model Enhancements

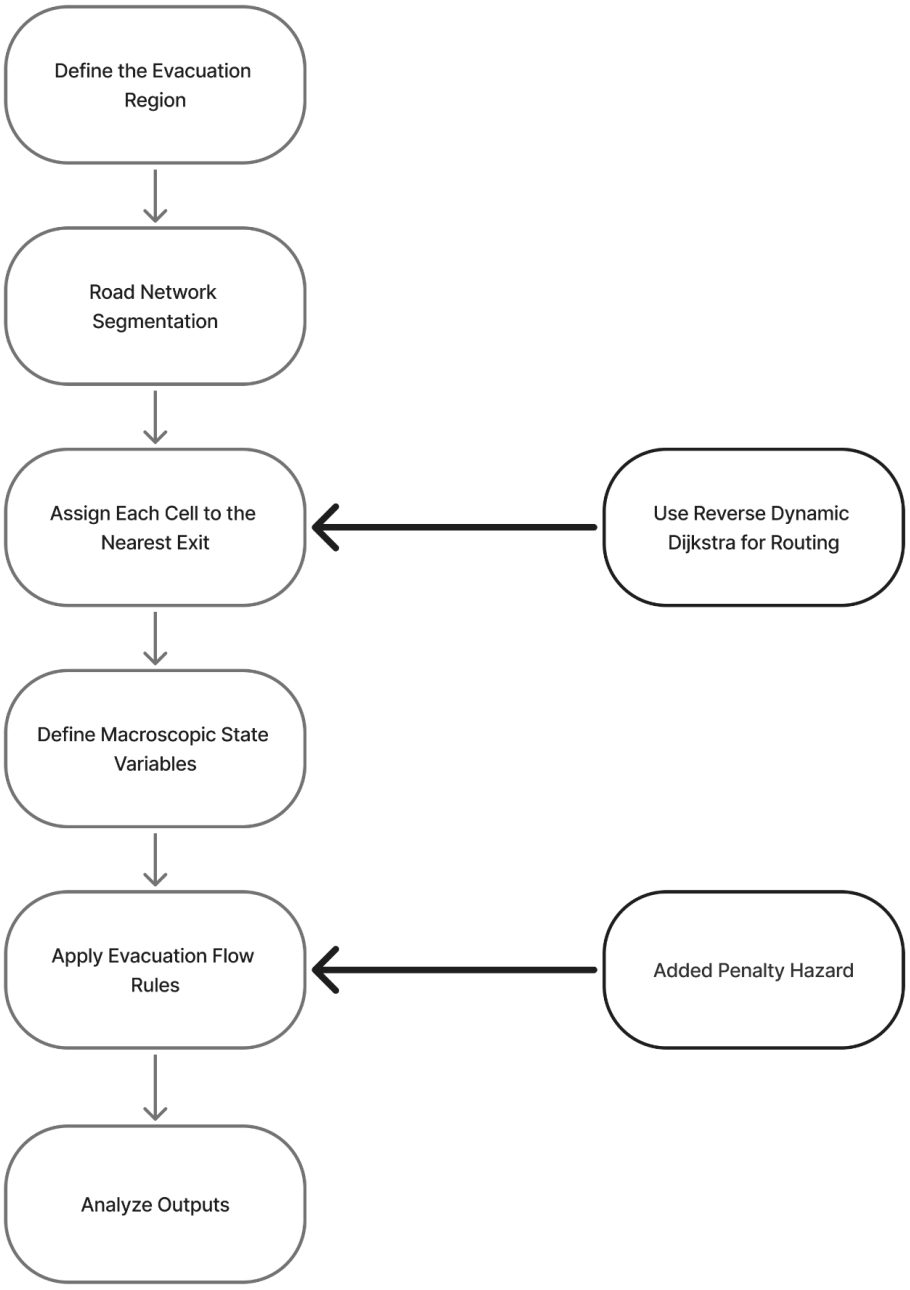

Figure 3.1. Mesoscopic Cellular Automata framework used in Lv et al.(2021) plus the added Reverse Dynamic Dijkstra routing and penalty hazard.

Figure 3.1 illustrates the baseline mesoscopic cellular automata (MCA) framework used for evacuation simulation, consisting of five main steps: defining the evacuation region, segmenting the road network into cells, assigning each cell to the nearest exit, defining macroscopic state variables, and applying evacuation flow rules. In this study, two components of this baseline workflow were modified to enhance routing accuracy and hazard responsiveness.

As shown in the figure 3.1, the "Assign Each Cell to the Nearest Exit" step is replaced by the integration of Reverse Dynamic Dijkstra, allowing the model to compute cost-aware evacuation routes rather than relying on static nearest-exit assignments. Similarly, the "Apply Evacuation Flow Rules" step is enhanced by incorporating hazard and congestion penalty values within each cell, enabling the system to dynamically adjust traversal costs when fire or congestion conditions arise. These two modifications preserve the original structure of the MCA framework while improving its ability to handle dynamic and hazardous evacuation environments.

3.3 Enhanced MCA Pipeline

This section outlines the end-to-end workflow of the Enhanced MCA model, including evacuation-region definition, road-cell construction, routing initialization, mesoscopic state updates, hazard-penalty integration, dynamic routing repair, and output extraction. The pipeline builds on the MCA framework of Lv et al. (2021) and incorporates Reverse Dynamic Dijkstra routing to enable adaptive, hazard-aware evacuation simulation.

3.3.1 Evacuation Region Definition

The enhanced MCA simulation begins by defining the evacuation region that serves as the spatial domain of the model. The **USTP Cagayan de Oro (USTP CDO)** campus is used as the primary map, while two additional university campuses serve as supplementary test environments to evaluate model generalizability. USTP CDO is selected because its spatial layout **fits the** mesoscopic mapping criteria described by Lv et al. (2021): it has a connected system of outdoor roads, walkways, and open traversal areas that can be segmented into mesoscopic road cells.

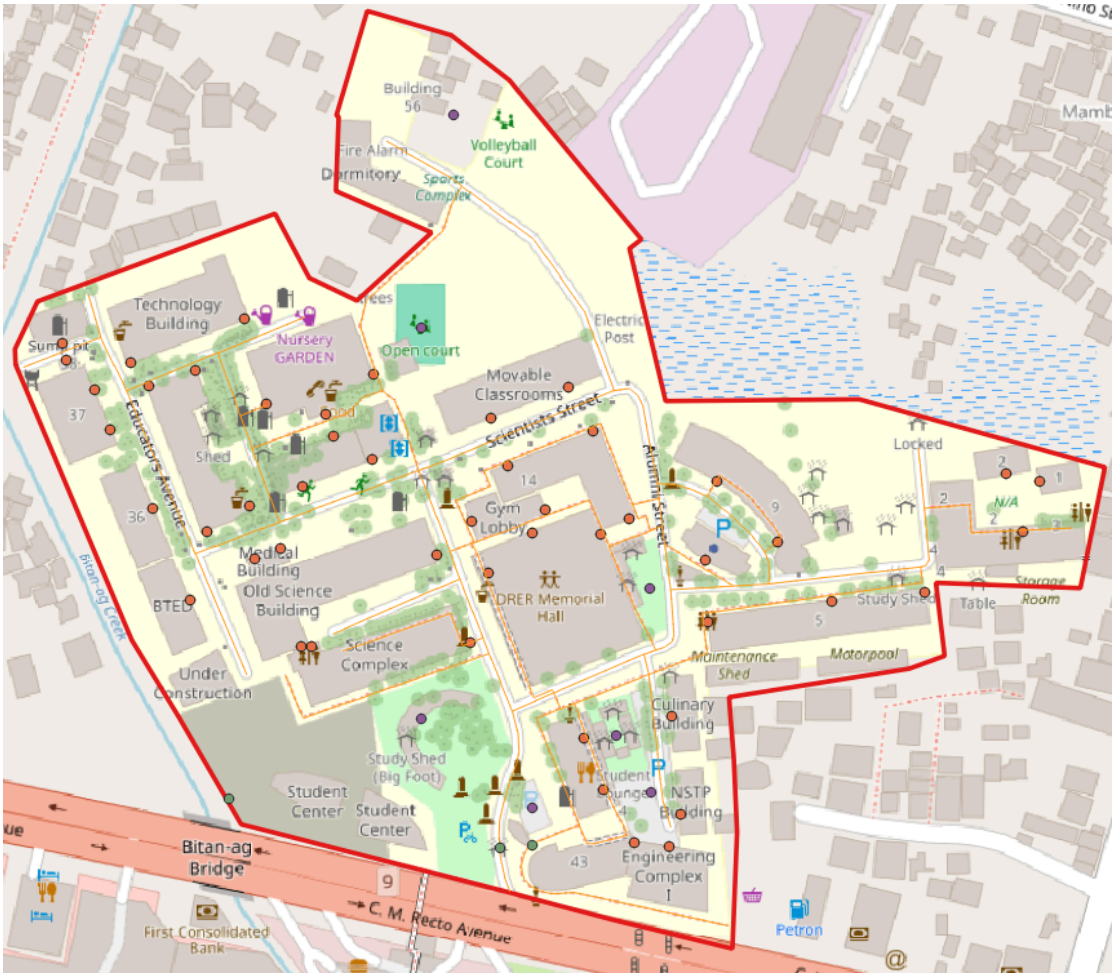

Figure 3.2. USTP CDO Campus mapped in QGIS software, including possible building exits, sink exits(entrances/exits), road networks(road cells).

In a mesoscopic map, only traversable pathways such as roads, footpaths, and other walkable areas are converted into meso-scale cells, while buildings function solely as source-loading areas. Following this principle, the USTP CDO map is outlined to include only its road-and-pathway network, consistent with the mesoscopic mapping approach of Lv et al. (2021). All non-walkable regions such as buildings, open fields, parking lots, and restricted spaces are excluded from traversal and serve only as contextual surroundings or loading zones. This produces a mesoscopic spatial structure suitable for routing, hazard modeling, and movement simulation in the enhanced MCA pipeline.

3.3.2 Road Network Segmentation and Road Cell Construction

Following the definition of the evacuation region, the next step is converting the campus pathways into a mesoscopic grid. Consistent with the mesoscopic principles of Lv et al. (2021), only walkable road-and-pathway segments, building entrances, entity spawn points, and campus entrances or exits are transformed into simulation cells, while building interiors remain non-traversable and act only as source-loading areas. In accordance with Lv et al.'s road-cell formulation, each mesoscopic cell represents a 10 m × 6 m road unit, providing sufficient spatial aggregation for crowd flow modeling while maintaining computational efficiency.

The USTP CDO campus pathways are processed into structured road cells by identifying all traversable segments, campus roads, footpaths, and open movement areas and dividing them into meso-scale road cells. This follows the same approach shown in Lv et al.'s road cell segmentation, where each pathway is split along its direction into rectangular cells that represent aggregated pedestrian flow rather than individual agents.

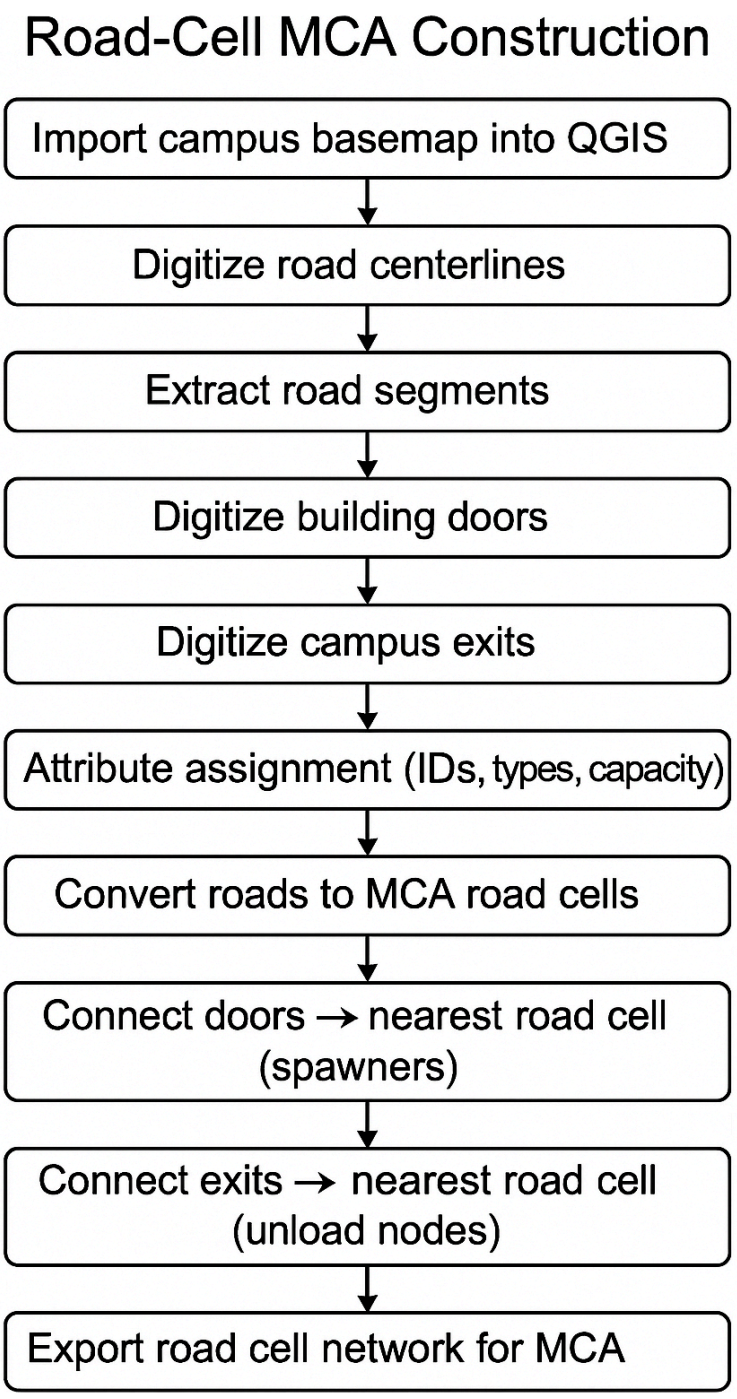

Figure 3.3. Workflow for Constructing the Road-Cell MCA Network from QGIS-Digitized Data

Each resulting road cell forms a basic unit of the mesoscopic grid and stores macroscopic variables such as density, speed, and flow during the

simulation. Building-adjacent cells serve as source-loading cells, mirroring the cell roles defined in the baseline model. The completed grid provides the structural foundation for routing initialization and hazard-aware movement in later steps of the enhanced MCA pipeline.

3.3.3 Routing Initialization Using Reverse Dynamic Dijkstra

After constructing the mesoscopic road network cells, the next step is to initialize the routing structure used by evacuees. In the baseline MCA model of Lv et al. (2021), each road cell is assigned to the nearest exit based solely on geometric distance, resulting in static exit choices that do not account for hazards or congestion.

To improve routing flexibility, this study replaces the nearest-exit allocation with a Reverse Dynamic Dijkstra approach. Instead of computing paths from every road cell to each exit, the algorithm begins at the exits and propagates outward across the grid, assigning each cell a traversal cost and an initial preferred direction. This reverse expansion is more computationally efficient, as it eliminates the need to compute thousands of source-to-exit paths individually (Oyola et al., 2017).

During initialization, traversal costs reflect only baseline factors such as road connectivity and geometric distance, while hazard and congestion penalties are incorporated later in the simulation. The result is a baseline routing field that identifies the most efficient exit under normal, hazard-free conditions (Lv et al., 2021). This field serves as the initial guidance structure

before dynamic penalties and local repairs adjust evacuee paths during the evacuation process.

3.3.4 Cell State Update with Hazard and Congestion Penalties(Core Foundation)

At each timestep, the state of every road cell is updated using the mesoscopic CA flow equations defined by Lv et al. (2021). Each cell maintains its number of evacuees $Ni(t)$, density $\rho i(t)$, speed $vi(t)$, and flow $Qi(t)$. These are updated using the standard MCA relationships:

Density update

$$\rho_i(t) = \frac{N_l(t)}{A_i} \qquad \textit{(Equation 3-1)}$$

Speed update (flow–density relationship)

$$v_i(t) = v_f(1 - \frac{\rho_i(t)}{\rho_{max}}) \qquad \textit{(Equation 3-2)}$$

Flow update

$$Q_i(t) = \rho_i(t) \cdot v_i(t) \qquad \textit{(Equation 3-3)}$$

Flow conservation

$$N_i(t + \Delta t) = N_i(t) + Q_{in}(i,t) - Q_{out}(i,t) \quad \textit{(Equation 3-4)}$$

Capacity constraint

$$Q_{out}(i,t) \leq C_i \quad \textit{(Equation 3-5)}$$

Where:

i = index of the road cell in the mesoscopic grid

t = discrete simulation time step

Δt = duration of one time step (set to 1 second)

$N_i(t)$ = number of evacuees in road cell i at time t

A_i = physical area of road cell i (m²)

$\rho_i(t)$ = pedestrian density in cell i at time t, computed as $Ni_i(t)/A_i(persons/m^2)$

$v_i(t)$ = mean walking speed of evacuees in cell i at time t (m/s)

v_f = free-flow walking speed in uncongested conditions (1.3–1.5 m/s)

ρ_{max} = maximum allowable pedestrian density in a cell (4–5 persons/m²)

$Q_i(t)$ = pedestrian flow through cell i at time t (persons/s)

$Q_{in}(i,t)$ = total inflow of evacuees entering cell i during $[t, t + \Delta t]$

$Q_{out}(i,t)$ = total outflow of evacuees leaving cell iii during $[t, t + \Delta t]$

C_i = capacity of cell iii, i.e., the maximum allowable outflow per time step, determined by cell width and ρ_{max}

These equations represent the baseline MCA dynamics and are used for every ordinary road cell in the network. Source-loading cells inject evacuees following the same loading rule in the original model, and exit cells remove evacuees from the system.

The enhanced model incorporates hazard and congestion effects by applying a penalty coefficient $C_{cell}(i,t)$ to the baseline MCA update. The penalty does not modify the original equations; instead, it serves as a multiplicative reduction applied to the effective speed and outflow:

Effective speed with penalty

$$v_i^{eff}(t) = v_i(t)\,(1 - C_{cell}(i,t)) \qquad \textit{(Equation 3-6)}$$

Effective flow with penalty

$$Q_i^{eff}(t) = \rho_i(t) \cdot v_i^{eff}(t) \qquad \textit{(Equation 3-7)}$$

Where:

$v_i^{eff}(t)$ = effective speed after applying hazard and congestion penalties. This value represents the reduced movement capability caused by fire, smoke, debris, terrain difficulty, or local crowding.

$Q_i^{eff}(t)$ = effective outflow after penalties, representing the actual number of evacuees that can move through cell i under hazardous or congested conditions.

$C_{cell}(i,t)$ = total hazard congestion penalty applied to cell i at time t, defined in Section 3.4.2 as the weighted sum of density, fire intensity, smoke, debris, terrain difficulty, and temporary obstructions

A higher penalty value results in a proportional decrease in local movement capacity, formally reflecting reduced traversability in hazardous or congested cells while keeping the MCA framework unchanged.

3.3.5 Evacuation Flow Simulation (Hazard Penalties | Enhanced Layer)

During the evacuation phase, the MCA model updates the macroscopic variables of each road cell density $\rho_i(t)$, speed $v_i(t)$, and flow $q_i(t)$ following the conservation and flow–density equations in Section 3.3.5. As evacuees move through the network, these values evolve over time, causing corresponding changes in congestion levels and hazard exposure.

At each simulation step, the hazard-aware penalty field

$$C_{cell}(i,t) = \alpha\rho_i(t) + \beta F_i(t) + \gamma S_i(t) + \delta D_i(t) + \epsilon T(r)_i + \zeta T(o)_i$$

Equation (3-8)

where:

ρ = normalized pedestrian density (0–1).

F = fire intensity (binary or continuous).

S = smoke density.

D = debris obstruction level,

and α, β, γ, δ, ϵ, ζ are tunable weight coefficients that represent the relative influence of each factor.

$T(r)$ = terrain difficulty modifier (based on slope, surface type)

$T(o)$ = temporary obstruction value (e.g., furniture, equipment, moveable barriers

is re-evaluated using the updated density, fire, smoke, and obstruction states. This update does not alter the MCA equations but simply refreshes the effective traversal cost of each cell based on current environmental conditions.

This introduces a hazard-aware extension to the original MCA model. Hazard Penalties are updated every 5 seconds. A 5 second update interval prevents noisy, unstable rerouting caused by minor second-to-second changes and instead captures only meaningful congestion, which typically develops over 3-7 seconds(Zhang et al., 2015). This interval also reduces computation

while keeping routing responsive, as recommended in mesoscopic flow models (Shi, Lee, & Ma, 2018).

3.3.6 Routing Update Trigger

The enhanced MCA framework continuously evaluates whether the precomputed routing field remains valid as conditions change during the evacuation. The routing field is updated only when an evolving hazard or congestion penalties cause the original Reverse Dijkstra labels to no longer represent the safest or most efficient direction. This condition occurs when a cell on the current route develops a significantly higher traversal cost due to fire, smoke, congestion, or other obstructions, or when a neighboring cell becomes more favorable than the previously assigned direction. When these situations arise, the model flags the affected region for local routing repair, which is carried out in Section 3.3.7.

3.3.7 Local Routing Repair When Conditions Change

Once a routing update is triggered, the model performs a localized correction to the Reverse Dynamic Dijkstra. The repair procedure updates the Reverse Dynamic Dijkstra labels within the impacted region by performing a localized relaxation process, ensuring that only the cells influenced by the new penalty values are recalculated. Evacuees then follow the updated routing field by selecting the neighboring cell with the lowest revised cost label, subject to the MCA capacity and flow constraints.

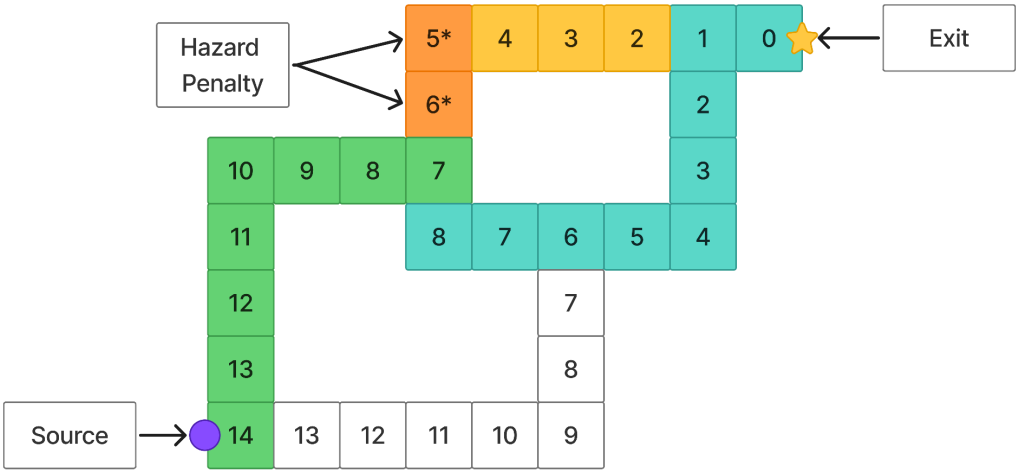

Figure 3.4 *Example of routing update and local repair. When hazard penalties appear on a previously optimal path*

The figure 3.4 shows how the routing field changes when a hazard appears on the originally optimal path. The green cells represent the initial safest route computed before any hazard occurred. When a hazard emerges on two critical cells (marked 5* and 6*), their traversal cost increases, causing the surrounding segment of the path (orange cells) to become unsafe and unusable. In response, the model triggers a routing update and performs a local repair around the affected region, adjusting only to the safest cell paths. This produces a new hazard-aware route, shown in teal, which reroutes evacuees safely around the penalized cells while still guiding them toward the exit.

3.3.8 Output Extraction and Evaluation

To ensure consistency with established mesoscopic evacuation modeling practices, the enhanced MCA model adopts the eight evaluation

metrics defined by Lv et al. (2021). These metrics serve as the basis for interpreting the model's statistical outputs, as shown in Table 1.

Table 1. Summary of evacuation performance metrics adapted from Lv et al. (2021).

Metric	How it is computed	Formula	Purpose	Result
Exit flow rate	Count the number of evacuees exiting per second via each exit cell. Computed from macroscopic inflow/outflow rules.	Flow = Density × Speed × Width	How many people leave through an exit each second.	Higher is better (too high may indicate dangerous compression)
Remaining evacuees over time	Sum evacuees still inside the evacuation network at each timestep.	Remaining = Sum of people in all cells	How many people are still inside the network at each time step.	Lower is better at each timestep
Density evolution in observed cells	Density computed for selected road, junction, exit, and source cells across time.	Density = People ÷ Cell Area	How crowded each road cell is.	Lower is better (avoids congestion and stampede risk)
Velocity of Evacuees	Speed obtained from density	Speed = Free Speed × exp(−Densi	People slow down as density increases.	Higher(closer to free-flow speed).

	using exponential congestion model.	ty ÷ Critical Density)		
Spatial distribution	For selected times (100–600 s), plot number of evacuees per cell.	Cell Load = People in each cell at selected times	Snapshot of which areas are crowded.	Balanced distribution is better; avoid heavy clustering at certain roads.
Exit Usage Distributio n	Total evacuees that passed through each exit (accumulat ed flow).	Exit Count = Total flow that passed through exit	How many people used each exit.	A more balanced distribution is better (prevents exit overload).
Total Evacuation Time	First timestep where remaining evacuees < 0.5.	Time ends when Remaining < 1 person	When the system has effectively evacuated.	Lower/sho rter is better.

3.4 Proposed Model Enhancements: Routing and Hazard Integration

This section introduces the enhancements that improve upon the baseline mesoscopic CA model of Lv et al. (2021): the Reverse Dynamic Dijkstra(RDD) algorithm, which provides global consistency and adaptively update cost-to-exit field, and the Hazard and Congestion Penalty formulation, which integrates density, smoke, fire, debris, terrain difficulty, and temporary obstruction into the routing and movement rules. These enhancements enable the MCA model to respond dynamically to evolving environmental

conditions and congestion patterns, producing more realistic and adaptive evacuation behavior (Oyola et al., 2017; Lv et al., 2021; Liu et al., 2023).

3.4.1 Reverse Dynamic Dijkstra Routing

The Reverse Dynamic Dijkstra (RDD) algorithm provides the global routing field used by the enhanced MCA model. While Section 3.3.3 introduces the initial labeling stage, this section describes the full routing mechanism, including global cost propagation, hazard-aware adjustments, and dynamic updates during simulation.

RDD initializes all exit cells with a distance of zero and assigns a large default value to the remaining traversable cells. Using a priority queue, the algorithm propagates distance labels outward from the exits and applies the relaxation rule,

$$dist(v) = min(dist(v), dist(u) \quad (\textit{Equation 3-9})$$

where:

- $dist(v)$ = the current cost-to-exit value assigned to cell vvv. This represents the estimated minimum evacuation cost from vvv to any exit.

- $dist(u)$ = the finalized or currently best-known cost-to-exit value of a neighboring cell uuu. Because RDD expands outward from exits, uuu is a cell whose label is already stable or recently updated.
- $w(u,\ v)$ = the traversal cost from cell u to cell v.
- $min(\cdot)$ = selects the smaller of the two values, ensuring that the routing field always reflects the lowest available evacuation cost.

This reverse expansion produces a global minimum-cost field, allowing every pedestrian to move by choosing the neighboring cell with the smallest distance value. Reverse labeling is particularly suitable for multi-exit evacuation because a single execution generates cost-to-exit values for the entire network simultaneously (Oyola et al., 2017).

As the simulation progresses, cell conditions change due to crowd density, fire intensity, smoke accumulation, debris, or terrain difficulty. These factors modify each cell's effective traversal cost through the hazard-penalty function like in *Equation 3-8*.

RDD incorporates these penalties by adjusting the effective traversal cost during relaxation, ensuring that the resulting routing field remains consistent with the current environmental conditions. When hazards appear along the previously optimal path, or when congestion or obstructions raise a cell's traversal cost, the algorithm identifies the affected region and performs a targeted update: only those labels that become inconsistent are reinserted into the priority queue and relaxed again. This behavior corresponds to the

dynamic component of RDD and allows the routing field to adapt without recomputing the entire network.

Through this mechanism, the enhanced MCA model maintains a routing field that continuously reflects both geometry and evolving hazard conditions. Pedestrians always move according to the globally minimal cost-to-exit values while the MCA flow rules impose capacity and density constraints during movement.

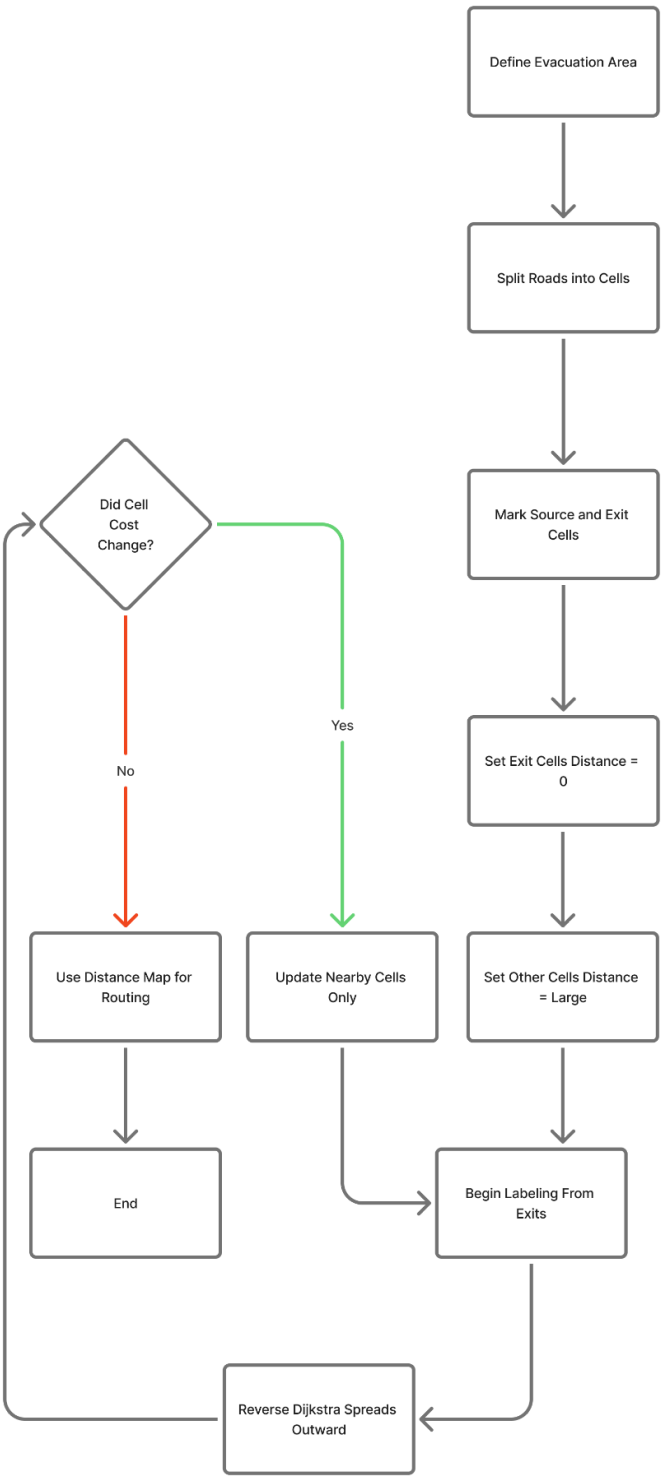

Figure 3.5 Reverse Dynamic Dijkstra flowchart

Time Complexity.

In terms of computational cost, the initial reverse labeling of RDD operates in $O((|V| + |E|)log|V|)$ using a heap-based priority queue, consistent with standard Dijkstra.

Dynamic updates throughout the simulation reuse the same data structures, allowing RDD to update only the necessary cells rather than recomputing the entire field, resulting in significantly lower routing overhead in hazard-changing environments.

```
Algorithm Reverse Dynamic Dijkstra(G, Exits)

PHASE 1 — INITIAL REVERSE LABELING
1:  For each cell v in G do
2:      dist[v] ← ∞
3:  End For
4:
5:  Create priority queue Q
6:  For each exit e in Exits do
7:      dist[e] ← 0
8:      Insert(Q, (0, e))
9:  End For
10:
11: While Q not empty do
12:     (du, u) ← ExtractMin(Q)
13:     If du > dist[u] then continue
14:
15:     For each neighbor v of u do
16:         w ← Cost(u, v)      # precomputed or static movement cost
17:         If dist[u] + w < dist[v] then
18:             dist[v] ← dist[u] + w
19:             Insert(Q, (dist[v], v))
20:         End If
21:     End For
22: End While


PHASE 2 — DYNAMIC UPDATE UNDER CHANGING CONDITIONS
23: For each simulation step t do
24:     Changed ← { cells whose traversal cost changed at time t }
```

```
25:
26:     If Changed is empty then continue
27:
28:     Create repair queue RQ
29:     For each u in Changed do
30:         Insert(RQ, (dist[u], u))
31:     End For
32:
33:     While RQ not empty do
34:         (du, u) ← ExtractMin(RQ)
35:         If du > dist[u] then continue
36:
37:         For each neighbor v of u do
38:             w ← Cost(u, v)    # updated local cost
39:             If dist[u] + w < dist[v] then
40:                 dist[v] ← dist[u] + w
41:                 Insert(RQ, (dist[v], v))
42:             End If
43:         End For
44:     End While
45: End For

RETURN dist[]
```

Algorithm Reverse Dynamic Dijkstra

3.4.2 Hazard & Congestion Penalty Formulation

To allow the routing field to adapt to evolving environmental conditions, each traversable cell is assigned a time-varying penalty value that represents its current level of congestion and hazard exposure. This penalty modifies the traversal cost used by the Reverse Dynamic Dijkstra (RDD) algorithm and adjusts the effective speed and flow in the MCA update rules. The combined penalty for a cell i at time t is expressed as in *Equation 3-8.*

3.4.3 Density Influence (α=0.5–1.0)

The density component follows mesoscopic CA studies such as Lv et al. (2021), who showed that pedestrian congestion is the most dominant

contributor to reduced speed and increased flow instability in meso-scale road-cell models. High $\rho_i(t)$ leads to slower movement and greater route competition, justifying a relatively strong coefficient range for α.

3.4.4 Fire Intensity (β=0.7–0.9)

Fire is treated as a severe hazard with high traversal cost. Deng et al. (2022) demonstrated that exposure to high-temperature zones reduces walking speed by 70–90% and substantially increases evacuation risk. This supports assigning a strong weight to $F_i(t)$, making fire-exposed cells significantly less desirable in the routing landscape.

3.4.5 Smoke Concentration (γ=0.4–0.6)

Smoke primarily affects visibility and breathable air quality. Liu, Li, and Sun (2023) reported that smoke can reduce movement efficiency by 40–60%, especially in enclosed or semi-open spaces. Because smoke is less damaging than direct fire but still significantly harmful, γ is assigned a medium-strength penalty.

3.4.6 Debris Obstruction (δ=0.15–0.25)

Debris narrows passable space and increases localized crowding. Yue et al. observed that even small obstacles reduce effective passage width and pedestrian throughput by 15–30%. Thus, debris contributes a moderate

penalty, increasing traversal cost in partially obstructed cells without overpowering the effects of fire or smoke.

3.4.7 Terrain Difficulty (ϵ=0.1–0.3)

Terrain slope and uneven ground influence walking speed. Aghabayk, Parishad, and Shiwakoti (2021) showed that sloped walkways reduce pedestrian speed by 10–30% depending on gradient. Because terrain effects persist throughout the evacuation but do not pose immediate danger, the coefficient is kept moderate.

3.4.8 Temporary Obstructions (ζ=0.1–0.2)

Movable barriers, fallen objects, and non-fixed obstacles can reduce flow efficiency by approximately 10–20%, particularly near narrow passages or intersections (Yue et al.). These obstructions warrant a light to moderate penalty because they impede movement but do not physically endanger evacuees.

Penalty fields are updated every 5 seconds to reflect evolving congestion and hazard conditions. This interval was selected as a balance between responsiveness and computational efficiency, based on pilot trials showing that shorter update periods provided negligible routing improvements while significantly increasing computational overhead.

The dynamic cost surface generated from these updated penalties is then passed to the Reverse Dynamic Dijkstra routing module, where it

directly influences path recalculation. As conditions worsen or improve across the network, the routing field adjusts accordingly, allowing evacuees to preferentially follow routes that are not only shorter but also safer and less congested. By integrating congestion and hazard feedback into the routing layer, the enhanced MCA framework captures the complex interactions between pedestrian density, fire propagation, smoke accumulation, and route accessibility, resulting in more adaptive and realistic evacuation behavior.

3.5 Experimental Design and Comparison Setup

This section describes how the proposed enhanced MCA model is evaluated against the baseline framework and alternative routing algorithms. All experiments are performed on mesoscopic road–cell grids derived from the selected university campuses, using identical initial conditions across model variants to ensure fair comparison.

3.5.1 Phase 1: Baseline MCA vs Enhanced MCA Without Hazard Penalties

In Phase 1, the goal is to compare the original Lv et al. mesoscopic CA framework against the enhanced MCA under neutral conditions, that is, without the influence of hazard penalties. This avoids biasing the results in favor of the enhanced model and isolates the effects of the routing and framework design.

Three configurations are considered:

1. Baseline MCA – Nearest-Exit Routing (Lv et al.).

   The control model follows the original mesoscopic CA setup:

road cells are segmented, buildings act as source-loading areas, and evacuees move according to the baseline density–speed–flow rules. Exit allocation uses nearest-exit or static subnetwork assignment, and no Reverse Dijkstra or hazard penalty is applied.

2. Enhanced MCA – Static Dijkstra Routing (No Penalties).

The enhanced MCA uses the same mesoscopic grid and flow-update rules but replaces nearest-exit assignment with a static Dijkstra shortest-path field from each cell to the exits. Hazard and congestion penalties are disabled in this phase, so routing depends only on geometric distance. This configuration represents a classical shortest-path improvement over the baseline while keeping the environment hazard-neutral.

3. Enhanced MCA – Reverse Dynamic Dijkstra (No Penalties).

The third configuration uses the full Reverse Dynamic Dijkstra (RDD) framework but with all penalty terms set to zero. In this setting, RDD behaves like a reverse shortest-path router with the ability to repair labels if congestion or movement dynamics introduce inconsistencies, but without fire, smoke, or debris penalties. Comparing this configuration with the baseline and static Dijkstra variants isolates the benefits of the enhanced routing structure itself, independent of hazard modeling.

Hazard penalties are deliberately disabled in Phase 1 to avoid a useless and biased comparison. If hazards were enabled only in the enhanced MCA, the model would be forced to react to additional risk variables (fire, smoke, debris), while the baseline MCA would ignore them entirely. In that case, any difference in performance would be confounded by model objectives rather than framework quality: the baseline would "look better" simply because it is not penalized for sending evacuees through dangerous cells. By keeping the environment hazard-neutral in Phase 1, the comparison focuses on structural and routing improvements nearest-exit routing versus static Dijkstra and hazard-free RDD under the same conditions.

3.5.2 Phase 2: Hazard-Aware Routing Comparison in Enhanced MCA

In Phase 2, the full hazard-aware enhanced MCA framework is activated, including the congestion and hazard penalty formulation described in Section 3.4. All configurations in this phase share:

1. The same mesoscopic road–cell grid,
2. The same fire and smoke evolution rules,
3. The same penalty field $C_{cell}(i, t)$ and
4. The same MCA density–speed–flow updates.

Only the routing algorithm is varied. Two routing strategies are compared:

1. **Reverse Dynamic Dijkstra (RDD, Proposed).**

   RDD computes a reverse cost-to-exit field from all exits and

updates it dynamically when penalty values change. Local repair is used to adjust labels only in affected regions, so routing remains consistent with evolving fire, smoke, debris, and congestion.

2. **Ant Colony Optimization (ACO) Routing.**

In the ACO configuration, multiple artificial ants explore candidate routes between sources and exits, updating pheromone levels based on path length and penalty cost. Evacuees then follow the most reinforced routes. This provides an alternative intelligent routing baseline that uses the same penalty information but a different optimization mechanism.

Standard Dijkstra is not included in Phase 2 because it is not designed for a fully dynamic hazard environment. Once hazard penalties become time-varying, a static shortest-path field computed at the beginning of the simulation cannot consistently reflect evolving fire, smoke, and obstruction patterns without full recomputation at each update step. This would either make the method unrealistic (if left static) or transform it into a different class of algorithm (if repeatedly recomputed), blurring the distinction from RDD. For this reason, Phase 2 focuses on routing methods explicitly suited to dynamic conditions RDD with local label repair and ACO with iterative pheromone updates under identical hazard-aware MCA dynamics.

3.5.3 Simulation Scenarios

Both phases are tested on the mesoscopic road networks of the USTP CDO campus (primary case study) and two additional university campuses (supplementary test sites). For each campus, synthetic scenarios are generated by varying:

1. **Exit configuration:** opening, closing, or adding synthetic exits and safe zones.
2. **Population size and distribution:** total number of evacuees and their allocation to different buildings.
3. **Source-loading patterns:** timing and rate at which evacuees are released from buildings into the road network.
4. **Hazard layout (Phase 2 only):** ignition locations, number of fire sources, and placement of debris or temporary obstructions along key routes.

Each scenario is run under all relevant configurations for its phase (Phase 1: baseline, enhanced + Dijkstra, enhanced + RDD; Phase 2: enhanced + RDD, enhanced + ACO), using the same initial map, exits, and population to ensure fair comparison. Results are averaged over multiple stochastic replications per scenario.

3.6 Implementation Plan

This section details how the enhanced MCA evacuation model will be developed, integrated, and executed. It presents the system architecture, required development tools, data preparation workflow, functional modules, testing strategies, and the expected outputs.

3.6.1 System Architecture Overview

The enhanced MCA evacuation model is implemented as a modular system composed of four main layers:

1. Data preparation layer where the QGIS-derived road-cell network, building doors, and campus exits are preprocessed and converted into MCA-compatible structures.
2. Routing Layer, which initializes baseline exit assignments using the Reverse Dijkstra algorithm and dynamically updates routing fields in response to hazards and congestion.
3. MCA Simulation Layer, responsible for applying mesoscopic state-update equations and propagating density, speed, and flow across the road-cell network
4. Output & Analytics Layer, which records evacuation metrics, visualizations, and per-timestep statistics for performance evaluation.

This modular architecture separates data handling, routing logic, dynamic hazard integration, and MCA flow updates, this ensures clarity, maintainability, and straightforward extension for future enhancements.

3.6.2 Development Tools

The MCA models will be developed using **Python 3.7+**, the latest stable release of the Python programming package. This development environment will be paired with the necessary supporting software and libraries listed in the following section.

Table 2. Development Tools and System Requirements

Category	Tool / Specification	Notes
Hardware Requirements	Modern CPU (quad-core or higher)	Required for multi-step MCA and routing simulations
	8–16 GB RAM	Ensures stable performance during hazard updates and array operations
	GPU (optional)	Only needed for visualization
Operating System	Windows 10 or newer	Stable environment for Python, QGIS, and map handling
Core Software	**P**ython 3.7+	Main simulation environment for the MCA model
	QGIS 3.x	Used to digitize road centerlines, building doors, and exits for Road-Cell MCA
Spatial Data Format	GeoPackage(.gpkg)	Stores road cells, building exits, safe zones, and university exits in a single spatial database
Python Libraries	GeoPandas	Loads and manages spatial layers from the GeoPackage

	Shapely	Performs geometric operations such as adjacency detection and proximity checks between road cells
	NetworkX	Constructs the road-cell flow
	Numpy	Handles numerical arrays and timestep-based updates of density and hazard variables
	Matplotlib	static and animated visualizations of evacuation dynamics
	SciPy	Used for numerical utilities, interpolation(optional), and future sensitive analysis.
	Pandas + OpenpyXL	Used to export structured simulation results to Excel (`.xlsx`) format for reporting and post-analysis

3.6.3 Data Preparation Workflow

The data preparation workflow transforms the raw campus map into the mesoscopic road-cell network required by the Enhanced MCA simulation. The workflow follows the mesoscopic principles of Lv et al. (2021), where only the walkable road network and loading/unloading points (building

doors and campus exits) are converted into simulation cells. The process consists of four major stages:

1. **Spatial Preprocessing in QGIS**

   The digitized campus map is imported into QGIS, where three essential geometries are created:

   a. Road centerlines (walkable network)

   b. Building doors (source-loading nodes)

   c. Campus exits (unloading nodes)

   Each layer is stored independently (e.g., road_segments.geojson, building_doors.geojson, campus_exits.geojson) to preserve clean structure and simplify attribute assignment.

2. **Road Segmentation**

   The road centerlines are split into uniform mesoscopic segments corresponding to the adopted road-cell dimensions (e.g., 10 m × 6 m, consistent with mesoscopic resolution used in Lv et al. (2021)).

   Each segmented polyline represents a future MCA cell with capacity and direction attributes.

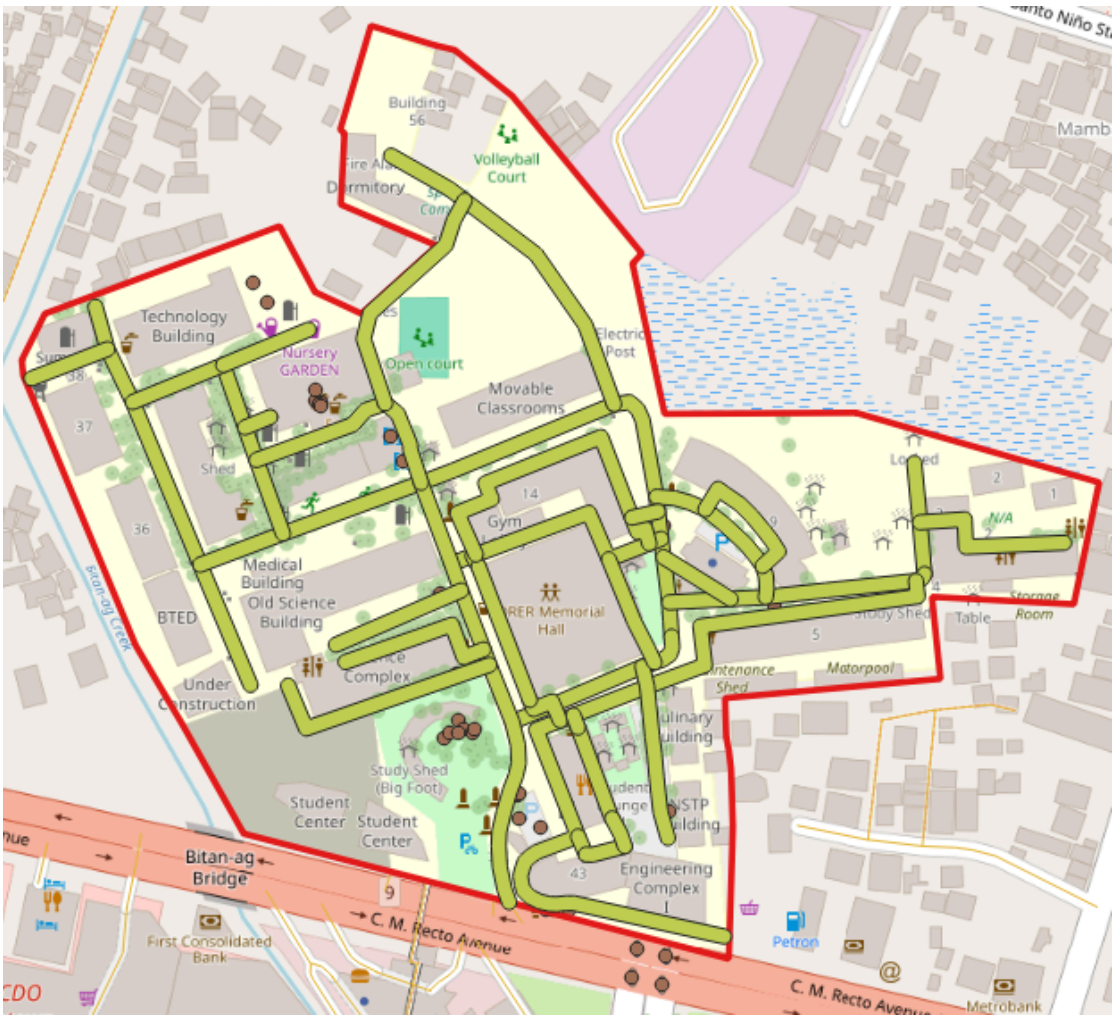

Figure 3.6. Illustrates the Campus map with 10m x 6m road cell segmentation following the configuration of Lv et al. (2021)

3. **Node Linking**

Building doors and campus exits are snapped to the nearest road cell:

a. Doors → nearest road cell(Source loading cells)

b. Exits → nearest road cell(unload nodes)

This step ensures that each evacuee initially transitions into the road network using a designated loading cell.

4. **Export for Simulation**

The processed layers are exported from QGIS in GeoJSON format and converted into MCA-compatible Python structures.

The output includes:

a. A list of road-cell nodes

b. Adjacency graph

c. The mapping of doors and exits to road cells

The final dataset serves as the structured input for the routing module and the full MCA simulation.

3.6.4 System Modules and Responsibilities

The Enhanced MCA evacuation system is organized into major models, each having their own responsibilities for a specific functional layer of the simulation workflow.

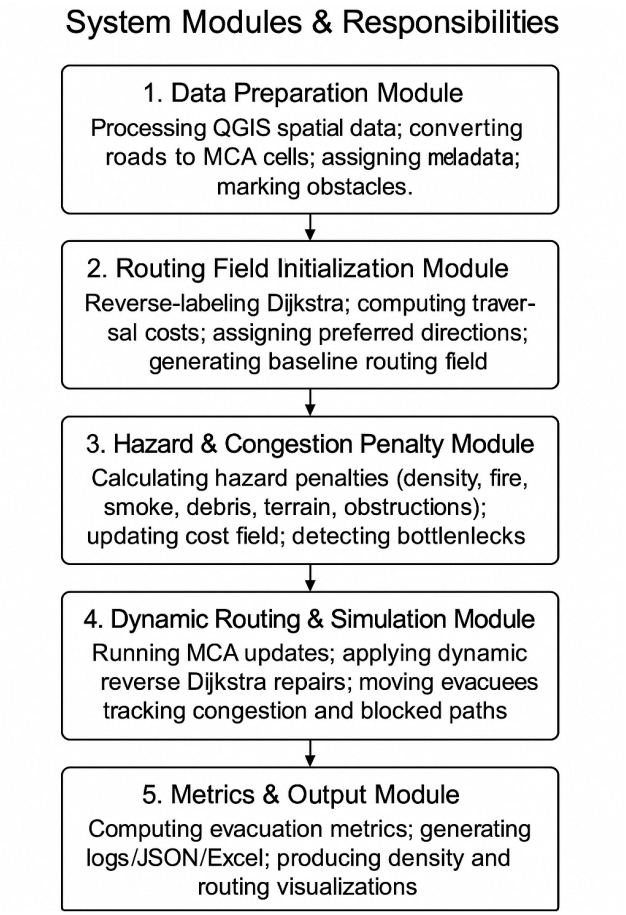

Figure 3.7. System Modules and Responsibilities.

Figure 3.6 illustrates the core modules of the proposed mesoscopic evacuation model:

1. **Data Preparation Module**

   This module imports QGIS-derived spatial data and converts roads, entrances, and spawn points into mesoscopic road cells(10 m x 6 m), attaching metadata such as connectivity and walkability. This follows MCA preprocessing standards where only traversable pathways are discretized (Lv et al., 2021).

2. **Routing Field Initialization Module**

   This module builds the baseline routing field using Reverse Dynamic Dijkstra, assigning each cell its initial cost-to-exit and

preferred direction under normal conditions. Reverse Expansions from exits reduces redundant source-to-exit queries and improves efficiency (Oyola et al., 2017; Lv et al., 2021).

3. **Hazard and Congestion Penalty Module**

   This module computes dynamic penalties—density, fire, smoke, debris, terrain difficulty, and temporary obstructions—scaled via coefficients, and detects congestion hotspots requiring repairs. Hazard-aware cost modification is consistent with multi-factor fields in recent evacuation literature (Liu et al., 2023).

4. **Dynamic Routing and Simulation Module**

   This module performs the MCA evacuation updates using mesoscopic continuity-based state rules and triggers Reverse Dynamic Dijkstra repairs when hazards disrupt existing routes. Cells update density, flow, and movement according to Lv et al.'s (2021) mesoscopic transition equations.

5. **Metrics and Output Module**

   This module calculates the MCA evacuation metrics found in Table X.1 and exports them into logs, JSON summaries and visual layers. Its metric design follows the standard performance metrics framework of Lv et al. (2021).

Each module outlines the key processes required to construct, update and evaluate the enhanced evacuation model.

3.6.5 Integration and Testing Plan

The integration and testing phase ensures that all system modules function as expected and that the Enhanced MCA model produces valid and reliable evacuation results. Testing covers correctness, stability, performance, and consistency with mesoscopic evacuation behavior described in Lv et al. (2021).

1. **Module Integration Strategy:**

   The system modules will be integrated in a pipeline-oriented manner, where each component passes its outputs directly to the next stage. Since the responsibilities of each module have already been defined in Section 3.6.4, integration will simply follow the established dependencies: data preparation feeds the routing module, routing initializes the hazard and congestion updates, and the simulation module produces the metrics consumed by the previous module. This approach ensures that modules remain loosely coupled, easy to update, and testable both individually and in sequence(Pressman & Maxim, 2020).

2. **Testing Phases**

   a. **Unit Testing**

      Each module(refer to Section 3.6.4) is tested independently. This verifies that core functions such as cost-field generation, hazard updates, and cell-state transitions produce expected outputs under controlled inputs (Pressman & Maxim, 2020).

b. Integration Testing

Modules are tested together following the pipeline found in Section 3.6.4. The goal is to confirm that outputs from one module correctly serve as inputs for the next module. Ensuring smooth data flow and functional compatibility across the system.

c. System/Scenario Testing

Full-scale evacuation scenarios(baseline, enhanced, standard dijkstra, ACO) are executed to assess end-to-end performance. Evaluations include correctness of routing behavior, stability under hazard-induced recalculation, and accuracy of performance metrics, consistent with validation practices in MCA evacuation studies (Lv et al., 2021)

3.6.6 Expected Outputs

The implementation of the enhanced MCA framework is expected to generate both simulation outputs and analytical artifacts. The system will produce:

1. A fully constructed mesoscopic road-cell grid, derived from QGIS datasets and converted into MCA-compatible structures.
2. Baseline and dynamically updated routing fields, showing optimal exits under normal and hazard-affected conditions.
3. Evacuation simulation results, including evacuee movements, density evolution, congestion behavior, and route changes.

4. Performance metrics, based on Lv et al. (2021), such as total evacuation time, peak density, exit load distribution, and bottleneck indicators.
5. Exported data products, including JSON logs, Excel summaries, and visualization layers (routing maps, density heatmaps, hazard-penalty fields) for analysis and presentation.

Together, these outputs demonstrate the behavior of both the baseline MCA and the proposed enhanced MCA framework, to which then will proceed to evaluating the Enhanced MCA model.

REFERENCES

1. Aghabayk, K., Parishad, N., & Shiwakoti, N. (2021). Investigation on the impact of walkways slope and pedestrians' physical characteristics on pedestrian normal walking and jogging speeds. Safety Science, 133, 105012. https://doi.org/10.1016/j.ssci.2020.105012

2. Burstedde, C., Klauck, K., Schadschneider, A., & Zittartz, J. (2001). Simulation of pedestrian dynamics using a two-dimensional cellular automaton. Physica A: Statistical Mechanics and its Applications, 295(3–4), 507–525. https://doi.org/10.1016/S0378-4371(01)00141-8
3. Deng, K., Zhang, Q., Zhang, H., Xiao, P., & Chen, J. (2022). Optimal emergency evacuation route planning model based on fire prediction data. Mathematics, 10(17), 3146. https://doi.org/10.3390/math10173146
4. Helbing, D., Farkas, I., & Vicsek, T. (2000). Simulating dynamical features of escape panic. Nature, 407, 487–490. https://doi.org/10.1038/35035023
5. Li, Y., Chen, M., Dou, Z., & Zheng, X. (2019). A review of cellular automata models for crowd evacuation. Physica A: Statistical Mechanics and its Applications. https://doi.org/10.1016/j.physa.2019.03.117
6. Liu, Y., Li, J., & Sun, C. (2023). Cellular Automaton model for pedestrian evacuation considering impacts of fire products. Fire, 6(8), 320. https://doi.org/10.3390/fire6080320
7. Lv, W., Wang, J., Fang, Z., & Mao, D. (2021). Simulation method of urban evacuation based on mesoscopic cellular automata. Journal of Mechanical Engineering, 70(10), Article 100706. https://doi.org/10.7498/aps.70.20210018

8. Mirahadi, F., & McCabe, B. Y. (2020). EvacuSafe: A real-time model for building evacuation based on Dijkstra's algorithm. Journal of Building Engineering, 32, 101687. https://doi.org/10.1016/j.jobe.2020.101687
9. Oyola, A., Romero, D. G., & Vintimilla, B. X. (2017). A Dijkstra-based algorithm for selecting the shortest-safe evacuation routes in dynamic environments (Short-Safe Evacuation Routes [SSER]). In Advances in Artificial Intelligence: From Theory to Practice (pp. 131–135). Springer. https://doi.org/10.1007/978-3-319-60042-0_15
10. Parisien, M.-A., Dawe, D. A., Miller, C., Stockdale, C. A., & Armitage, O. B. (2019). Applications of simulation-based burn probability modelling: A review. International Journal of Wildland Fire, 28(12), 913–926. https://research.fs.usda.gov/treesearch/60727
11. Patac, J. C. J., & Vicente, A. J. O. (2019, November 14–15). Urban fire spread modelling and simulation using cellular automaton with extreme learning machine. In The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (Vol. XLII-4/W19). PhilGEOS x GeoAdvances, Manila, Philippines. https://doi.org/10.5194/isprs-archives-XLII-4-W19-319-2019
12. Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education. https://www.researchgate.net/publication/336251012_Software_Engineering_A_Practitioner's_Approach_9th_Edition
13. Qiu, F., Wang, J., Zhang, T., & Li, P. (2024). Effects of timestep granularity on mesoscopic evacuation simulation. Physica A: Statistical

Mechanics and its Applications. https://www.mdpi.com/2571-6255/7/3/100

14. Shahabi, K., & Wilson, J. P. (2014). CASPER: Intelligent capacity-aware evacuation routing. Computers, Environment and Urban Systems, 46, 12–24. https://doi.org/10.1016/j.compenvurbsys.2014.03.004
15. Shi, M., Lee, E. W. M., & Ma, Y. (2018). A novel grid-based mesoscopic model for evacuation dynamics. https://doi.org/10.1016/j.physa.2017.12.139
16. Stevens, S., & Rush, D. (2025). Urban fire spread modelling: A review of dynamic computational models and potential for application to informal settlement fires. International Journal of Disaster Risk Reduction, 110, 105528. https://doi.org/10.1016/j.ijdrr.2025.105528
17. Tordeux, A., Lämmel, G., Hänseler, F. S., & Steffen, B. (2018). A mesoscopic model for large-scale simulation of pedestrian dynamics. Transportation Research Part C: Emerging Technologies, 93, 265–285. https://doi.org/10.1016/j.trc.2018.05.021
18. Villamor, F., & Goldman, R. (2017, February 8). Fire tears through Manila slum, leaving 15,000 homeless. The New York Times. https://www.nytimes.com/2017/02/08/world/asia/fire-tears-through-manila-slum-leaving-15000-homeless.html
19. World Health Organization. (2023, October 13). Burns. https://www.who.int/news-room/fact-sheets/detail/burns

20. Xu, D., Huang, X., Mango, J., Li, X., & Li, Z. (2020). Simulating multi-exit evacuation using deep reinforcement learning. arXiv Preprint, arXiv:2007.05783. https://doi.org/10.48550/arXiv.2007.05783
21. Yue, Z., Ma, Z., Yao, D., He, Y., Gu, L., & Jing, S. (2025). Beyond static estimates: Dynamic simulation of fire–evacuation interaction in historical districts. Applied Sciences, 15(12), 6813. https://doi.org/10.3390/app15126813
22. Zhang, J., Wang, Y., Wang, R., & Seyfried, A. (2015). Pedestrian dynamics in bottleneck flow. https://iopscience.iop.org/article/10.1088/1742-5468/2011/06/P06004
23. Zhu, D.-D., & Sun, J.-Q. (2021). A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute. IEEE Access. Advance online publication. https://doi.org/10.1109/ACCESS.2021.3053169