Chapter 7

Indoor emergency evacuation system using IoT-enabled WSNs

7.1 Introduction

In emergency evacuation systems, the Internet of Things (IoT) plays a major role in planning escape paths for trapped individuals. Sensor nodes are integral parts of any IoT-based system. Energy-limited sensor nodes are deployed in the monitoring environment to monitor different physical parameters such as temperature, fire, smoke, and humidity [148]. Deployed sensor nodes form a Wireless Sensor Network (WSN) to assist any IoT-based application. IoT-enabled WSN is mainly responsible for collecting real-time information from the deployed sensor nodes [149]. Recently, IoT found its application in many fields like Industry 5.0, smart cities, healthcare, home automation, surveillance, and emergency evacuation systems. [150, 151, 152, 153, 154].

Most public infrastructures have poor map planning and high individual density, which leads to many casualties and economic losses in an emergency situation. Therefore, self-evacuation by trapped persons tends to be more blind due to a lack of effective and safe navigation path guidance. In some cases, many deaths occur because of wrong escape decisions, clogging of individuals, and panic [155]. Furthermore, this also leads

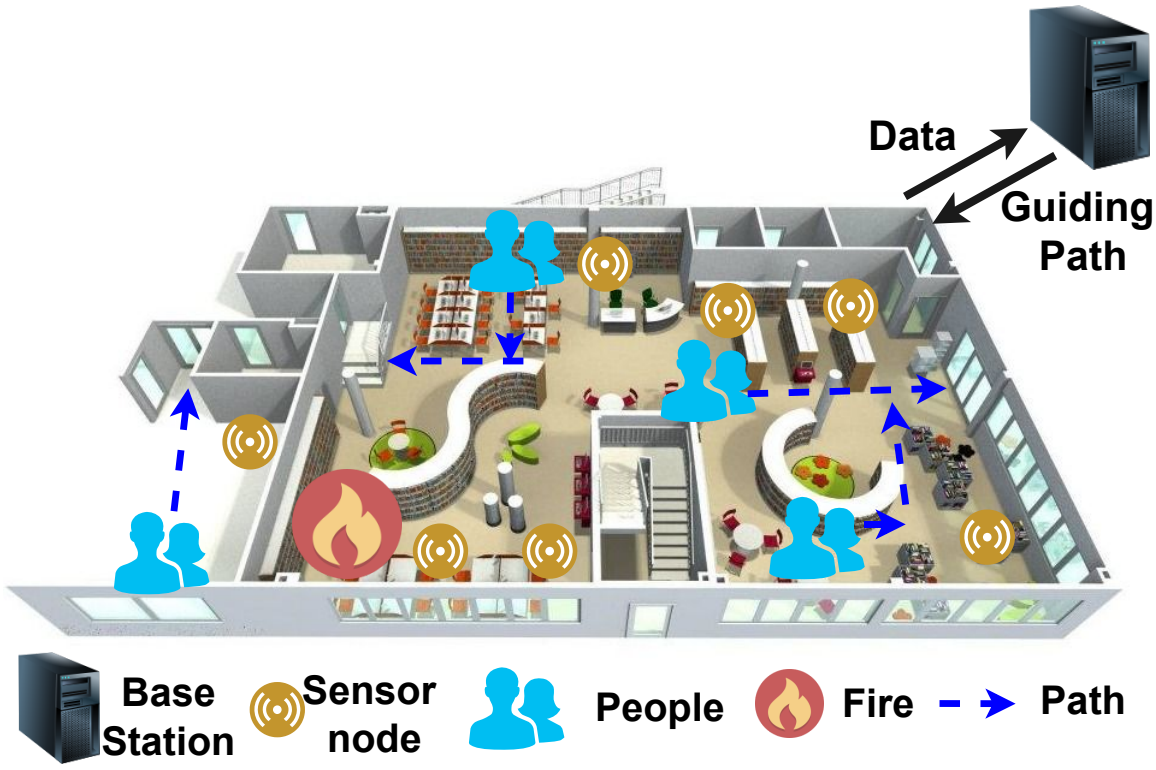


Figure 7.1: Emergency evacuation system.

to secondary stampede accidents. During a fire emergency, low visibility and abnormal communications occur due to high temperatures and heavy smoke in public infrastructures. In different areas, the severity and number of trapped persons are different, which leads to trouble in evacuation. Existing emergency evacuation systems [156, 157] only focus on static fire conditions. They do not consider the change in path planning with the change of fire region. Thus, it is important to plan a safe evacuation path for every individual by considering the change in the fire shape with time to avoid large detours during evacuation.

The existing evacuation systems do not sense and assess dynamically changing fire situations. In a mass evacuation, existing emergency evacuation systems do not deliver satisfactory assistance because of high variation in the fire region with time. Two types of systems are majorly used in emergency evacuation [158]. One is an evacuation system based on the social behaviour modelling of the masses [159]. Another is

an evacuation system based on enhanced shortest path planning [160], [161]. Recently, passive evacuation has been the result of many emergency evacuation systems. Evacuation path optimization has become the primary goal of almost all emergency evacuation systems. In the existing systems, the shortest geometric route is selected as the optimal evacuation path based on the two-dimensional plane of the public infrastructure. However, the existing systems do not consider the spread of fire with time and the shift of the psychological state of people on change in the navigation path. Therefore, the existing systems are compromised in safeness and efficiency. This leads to increased secondary accidents, large detours, panic in individuals, and deaths. Therefore, this chapter proposes a Dynamic emergency Evacuation system for Shortest-Safe path Navigation (DESSN) using IoT-enabled WSNs for smart buildings. The proposed DESSN approach dynamically adjusts the path and creates the shortest safe route for individuals in an emergency. The DESSN approach considers the future fire spread scenario in the region and designs safe paths for evacuees accordingly. It also avoids significant detours and trapping when an emergency arises in any public infrastructure with the help of IoT-enabled WSNs. Precisely, the following features are presented in the proposed dynamic emergency evacuation system.

1. **Safe Path:** The base condition for safety is that the path followed by evacuees must not contain any fire/hazardous node. In the proposed work, a safe route is planned for each evacuee.

2. **Consider future fire shape:** Path planning algorithm also considers the continuous changes in the fire region, such that the path found is safe for current as well as future conditions.

3. **Short Path:** The path found by the proposed emergency evacuation system is the shortest, which evacuates more evacuees in the least possible time.

4. **Scalable and Fast:** The proposed system is scalable to large networks and

efficiently assists in real-time emergency evacuation.

After installing IoT-enabled WSNs in public infrastructure/smart buildings, the proposed algorithm accomplishes all the above-mentioned features. Figure 7.1 shows the fire evacuation system. Every sensor node is attached to a heat/smoke detector for fire detection and intelligent navigation light to guide the evacuation path. Furthermore, a buzzer is connected to each sensor to alert all individuals in emergency situations. The following are the major contributions of this chapter.

1. This chapter proposes the DESSN approach that finds a dynamically adjusted shortest-safe path for individuals during an emergency.

2. This chapter finds the shortest safe evacuation path using a $fireMap$ and a $routeMap$.

3. Extensive simulations have been performed to test the performance of the proposed DESSN approach. The simulation results show that the proposed approach outperforms the existing approaches.

7.2 System Model

Figure 7.2 shows a typical emergency evacuation scenario for public infrastructure. It is surrounded by walls and contains one or more individuals and multiple exits. Purple boxes symbolize individuals that need to be evacuated from this infrastructure. An indoor occupancy detection system [162] is used for locating evacuees. The system guides these individuals along a short, secure path towards the exit door. Red and white boxes symbolize areas that are monitored by sensor nodes. Red boxes indicate the fire region, and white boxes represent the safe part. Purple lines depict potential navigation paths towards the exit. One Base Station (BS) is located outside the infrastructure for communicating with all the installed sensor nodes.

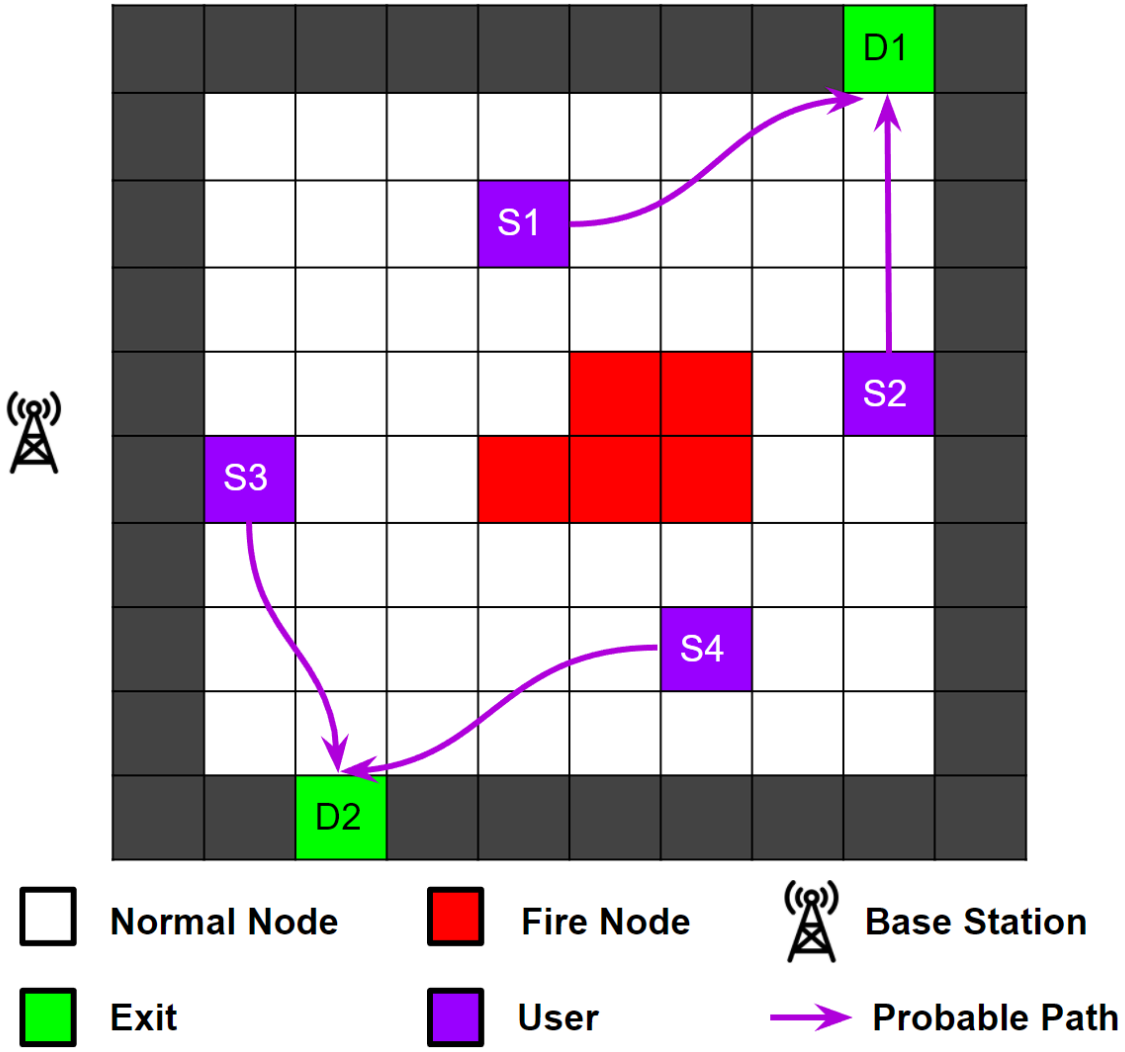


Figure 7.2: Emergency evacuation system in a typical public infrastructure.

7.2.1 Network model

A matrix of size $r \times c$ is formed based on the installed wireless sensor network structure where r represents the number of rows, and c represents the number of columns. The value of r and c depends on the monitoring infrastructure size. The size of each cell depends on the sensing range of sensor nodes. This work considers the size of each cell as 10 m^2. There is one sensor node present in every cell. This matrix is defined as a graph G, which contains a set of vertices V and a set of edges E. $ngb(a)$ represents the set of neighboring vertices of any vertex $a \in V$. G, E, V, and $ngb(a)$ are represented

Table 7.1: Notations and descriptions

Notations	Descriptions
d	Distance between source and destination node
G	Graph of network
V	Set of vertices
E	Set of edges
a	Vertex in set V
$ngb(a)$	Set of neighboring vertices of vertex a
e_{ab}	Edge connecting vertices a and b
S	Source or Evacuee
D	Door or Exit
F	Fire region
W	Wall
K	Set of fire spread rate/speed
T	Time when a fire will reach all the exit nodes

by the following equations:

$$G = (V, E) \tag{7.1}$$

$$E = \{e_{ab} = (a, b) \mid a, b \in V\} \tag{7.2}$$

$$V = \{a_i \mid i \in [1, r \times c]\} \tag{7.3}$$

$$ngb(a) = \{b \mid (a, b) \in V,\ e_{ab} \in E\} \tag{7.4}$$

Graph G can have a maximum of N vertices, where $N = r \times c$. This case will occur when no wall is present in the given network. So, the total number of vertices $|V|$ is represented by the following equation:

$$|V| < N \tag{7.5}$$

The installed wireless sensor network is converted into a rectangular matrix of size $r \times c$ where an undirected edge is present between every pair of neighbouring cells. A cell has eight neighbours. The cells present on the top, bottom, left, and right are considered neighbours. The cells diagonally adjacent to the current cell are also considered neighbours. Equation 7.6 [163] calculates the total number of edges $|E|$. Since there is at least one vertex in the network, hence values of r and c will always be greater than or equal to 1. Therefore, the rest of the terms apart from $4rc$ can be ignored in the

third step.

$$\begin{aligned} \therefore |E| &= (4(r-2)(c-2) + 5(r-2) + 5(c-2) + 6) \\ \implies |E| &= 4rc - 3(r+c) + 2, \ (\because r, c \geq 1) \\ \implies |E| &< 4rc, \ (\because N = r \times c) \\ \implies |E| &< 4N \end{aligned} \tag{7.6}$$

Table 7.1 contains the description of different notations used in this chapter.

7.2.2 Assumptions

For developing the proposed system, below mentioned assumptions are considered:

- There is a uniform distribution of all the sensor nodes in the wireless sensor network.
- Static sensor nodes with limited energy are deployed in the public infrastructure to monitor the emergency situation.
- In this work, only one Base Station (BS) is used with high computational power and unlimited energy.
- Sensor nodes send data to the BS through multi-hop communication. This work uses the Time Division Multiple Access (TDMA) protocol to mitigate interference [164].

7.3 Proposed Scheme

The proposed system architecture is divided into two units such as the software unit and the hardware unit. The hardware unit consists of the hardware components of the IoT-enabled WSN. Every sensor node takes input data from the current scenario

and sends this data to the software unit. The software unit runs on the BS/Host server, which processes this data. It calculates the safe and shortest evacuation path for every individual using the proposed path planning algorithm. Furthermore, it sends the resulting action information to every sensor node to guide evacuees toward the exit.

7.3.1 Software unit

The entire public infrastructure's map is simulated as a matrix at the BS/Host server based on the installed WSN. Sensor nodes detect fire and send this data to BS. Once BS receives the current input scenario from the hardware unit, the software unit processes the real-time received data by transforming it into a matrix. The input matrix shows the current fire situation. Different cell values are used to represent fire regions, free paths, exits, evacuees, and walls. The input matrix is updated based on the data received from the sensor nodes. Now, the proposed algorithm is executed in the BS/Host server and constructs a *fireMap* matrix. Each cell updates its information when the fire reaches that cell. A *routeMap* matrix is constructed that stores the shortest paths to the exit for each future time instance. The optimal shortest-safe path is calculated for each evacuee, considering the current and future changes in the shape of the fire region. This optimal route direction is sent to each sensor node to guide the evacuees toward the exit. Python language is used to implement this whole algorithm, and the Tkinter library was used to create a graphical user interface of simulations.

7.3.2 Hardware unit

Figure 7.3 illustrates the proposed hardware unit architecture and its components. A WSN is installed in the monitoring public infrastructure. A combination of sensors is used to detect fire. Each sensor node in this network has the following major components:

1. **D1 Mini (ESP8266):** D1 Mini is an integrated ESP8266 based WiFi-enabled

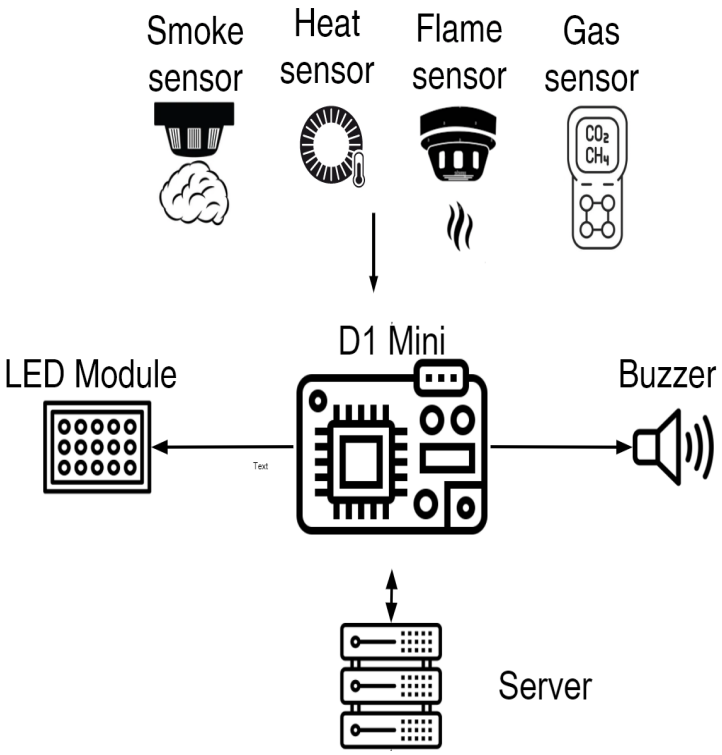


Figure 7.3: Hardware module.

microcontroller with 11 digital input/output pins and one analogue input pin (max 3.2 V). This microcontroller will process all the attached sensor's data.

2. **Flame Sensor:** Flame sensor detects the occurrence of fire or flame in the environment.

3. **Smoke sensor:** It monitors the environment to detect the smoke.

4. **Gas sensor:** It detects the presence of combustible gasses.

5. **Heat sensor:** It monitors the environment to detect the increase in heat level.

6. **MAX7219 LED Module:** It provides an LED display of 8×8 pixels. This module is used to navigate each evacuee in the correct direction.

7. **Buzzer:** If the fire is detected in any region, the buzzer alerts all the individuals about the occurred emergency.

7.3.3 Shortest safe path planning algorithm

After the input matrix is obtained at the BS/Host server, finding the shortest-safe path for each evacuee is performed in the following three stages.

7.3.3.1 Building the fire map

A $fireMap$ is a matrix in which each cell value contains the time when the fire will reach that cell. Each cell's value is initialized with an infinity value in this matrix. Furthermore, the Dijkstra algorithm is applied to calculate the time for the fire to reach each cell. Fire spread factor K is calculated by dividing the distance between two neighbour sensor nodes with the time difference in the detection of fire at these nodes. If locations of i^{th} and j^{th} sensors are (x_i, y_i) and (x_j, y_j), then *K[j]* is calculated as follows.

$$K[j] = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{t_i - t_j} \tag{7.7}$$

where i^{th} node is source of fire, t_i and t_j are the time when fire is detected at i^{th} and j^{th} node. A $fireMap$ is constructed by initiating Dijkstra's algorithm from the fire nodes. $fireMap$ is a matrix of size $r \times c$, with r rows and c columns. Thus, for a vertex $a_i = \{x_i, y_i\}$, time remaining for fire to reach this cell is $fireMap[x_i][y_i]$.

Algorithm 9 shows the pseudo-code of building $fireMap$. Initially, the priority of all *fire vertices* is set to zero. Next, all *fire vertices* are added to the priority queue Q. A vertex b with minimum priority is removed from Q, and its priority value is added to *fireMap*. *fireTime* and priority p is calculated for each unexplored adjacent vertex of b, and it is added into Q. *fireTime* is calculated by dividing the distance between the current node and the parent node by the fire spread factor K. For nodes where fire has not spread yet, the fire spread factor is the same as their parent node. Priority p is the sum of the current vertex's *fireTime* and the parent vertex's priority. These steps are repeated until Q becomes empty.

Algorithm 9: Fire Map building algorithm

Input: Current scenario graph/matrix: G
Output: Fire map: $fireMap[]$

```
1  Q:= declare a priority queue
2  for each vertex a ∈ G do
3      if a is Fire vertex then
4          explored[a] ← true
5          p[a] ← 0
6          Insert a in Q with priority p[a]
7  while Q ≠ φ do
8      n ← remove node with minimum priority from Q
9      b ← n.vertex
10     fireMap[b] ← p[b]
11     for each a adjacent to b do
12         dis ← length(a, b)
13         fireTime = dis/K[b]
14         if not explored[a] then
15             explored[a] ← true
16             Insert a in Q with priority p[b] + fireTime
17 return fireMap
```

7.3.3.2 Building route maps

Route maps are a series of matrices that store the shortest path traversal from any node to the exit for every time instance $t \in [0, T]$. These are built by applying a modified Dijkstra's traversal from exit nodes for every time instance t ($t \leq T$). It will stop at $t = T$, which indicates the current scenario.

A *routeMap* is constructed by initiating Dijkstra's algorithm from the exit nodes at different time instances. Dijkstra is an uninformed algorithm with very low space and time complexity compared to other shortest-path algorithms such as Bellman-Ford and Floyd-Warshall. $routeMap$ is a matrix of size $r \times c \times T$, where r is the number of rows, c is the number of columns, and T is the time till the fire reaches all the exit nodes. Therefore, for any individual currently standing at vertex $a_i = \{x_i, y_i\}$, it is possible to escape from the emergency if $routeMap[x_i][y_i][t] > 0$ for some $t \in [0, T]$.

Algorithm 10 shows the pseudo-code of building route maps. Initially, all exit ver-

Algorithm 10: Route Map building algorithm

Input: Current scenario graph/matrix: G
Output: Route maps: $routeMaps[][]$

```
1  for t from 0 to T do
2      Q:= declare a priority queue
3      for each vertex a ∈ G do
4          if fireMap[a] > t & a is Exit vertex then
5              routeMaps[a][t] ← t
6              explored[a] ← true
7              Insert a in Q with priority t
8      while Q ≠ φ do
9          n ← remove node with maximum priority from Q
10         b ← n.vertex
11         curT ← n.priority
12         for each a adjacent to b do
13             reachT ← routeMaps[b][t]− TravelTime(a, b)
14             if not explored[a] & fireMap[a] > reachT then
15                 routeMaps[a][t] = reachT
16                 explored[a] ← true
17                 Insert a in Q with priority routeMaps[a][t]
18         if explored[c] is false for some c ∈ ngb(b) then
19             Insert c in Q with priority curT − 1
20 return routeMaps
```

tices are added into *routeMap*. A vertex b with maximum priority is removed from Q. For each adjacent vertex a of vertex b, reach time $reachT$ is calculated by subtracting the travel time from a to b to *routeMap* value of b. If vertex a is unexplored and reach time $reachT$ is less than *fireMap* then $reachT$ is added to *routeMap*. a is inserted to Q with priority *routeMap[a][t]*. These steps are repeated until Q becomes empty.

7.3.3.3 Selecting optimal evacuation path

The proposed algorithm is based on the presumption that the shape of the fire region is continuously changing, and it can spread in the future. So, a short path planned for the current fire scenario will change for the user due to the changes in fire shape. Therefore, the user needs to backtrack or follow a different path to the exit, which

leads to large detours and even permanent trapping in the hazardous region. Thus the main objective of this algorithm is to plan an optimal route for current and future fire scenarios. The shortest path is found for each evacuee, which is safe in current as well as future fire scenarios. This path is computed for every individual by applying a binary search through $routeMap$ from time $t = 0$ to $t = T$ till it finds a valid path. Considering future fire scenarios, this selected path is the shortest and safe path.

7.3.4 Space and time complexity

The space and time complexities of the proposed DESSN algorithm are $O(K.N)$ and $O(K.N \log(K.N))$, respectively. N is the maximum possible size of the matrix (network). Here $N = r \times c$. The proposed DESSN algorithm consists of three stages. The first stage is to create the $fireMap$. The second stage is to construct the $routeMap$. The third stage is to select the optimal shortest-safe evacuation path for each evacuee. Moreover, the time and space complexities for each stage are as follows.

***Lemma 1*:** The time and space complexities of the $fireMap$ creating stage are $O(N \log N)$ and $O(N)$.

***Proof*:** The proposed DESSN approach uses the Dijkstra algorithm to build the $fireMap$. The worst-case time and space complexities of the Dijkstra algorithm are $O((E + V) \log V)$ and $O(V)$, respectively. Here, the number of edges is E, and the number of vertices is V. Since $E = N$, and $V = N$ from equation 7.5 and 7.6 the time complexity is $O((4N + N) \log N) \approx O(N \log N)$ and space complexity is $O(N)$.

***Lemma 2*:** The time and space complexities of the $routeMap$ creating stage are $O(K.N \log(K.N))$ and $O(K.N)$.

***Proof*:** The $routeMap$ construction stage in DESSN implements a modified version of Dijkstra's algorithm where a particular node is reinserted in the priority queue until all its valid neighbour nodes get visited. Suppose the fire spread rate is K m/s, then the maximum iterations for visiting all the neighbouring vertices of any vertex will be

K. So the effective number of vertices that can be taken in this modified Dijkstra is $K \times V$, while the number of effective edges will remain the same. Therefore, the worst-case time complexity of the $routeMap$ construction stage is $O(K.N \log(K.N))$ and the worst-case space complexity is $O(K.N)$, for $V = E = N$.

***Lemma 3*:** The time and space complexities of selecting the optimal evacuation path for each evacuee are $O(N + \log T)$ and $O(N)$.

Proof: There are two steps involved in finding the optimal route for an evacuee. The first step is to select the $routeMap$ with the minimum evacuation time for the evacuee's starting coordinates. This is done by applying a binary search from time $t = 0$ to $t = T$ for all the route maps, which takes $O(\log T)$ time and $O(1)$ space complexity. The second step is to backtrack on the selected route maps to get the whole evacuation path, which takes $O(N)$ time and $O(N)$ space complexity. Therefore, the time and space complexities of both steps are $O(N + \log T)$ and $O(N)$, which are the total time and space complexities for finding the optimal evacuation path for each evacuee.

7.4 Simulation Results and Analysis

Simulations of the proposed algorithm are performed by implementing the algorithm in Python. The Tkinter library is used to display the visual user interface of the simulation. In each simulation, an input matrix of size $r \times c$ (where r and c are integers) is passed for the initial time instance. The Fire Dynamic Simulator [165] is used for simulating the dynamic spread of fire. The walking speed of the crowd follows the Gaussian distribution with a mean value of 1.34 m/s [166]. Hence, in these simulations, the average moving speed of an evacuee is taken as 1.34 m/s. The performance of the proposed DESSN algorithm is evaluated in different simulation scenarios, which are as follows.

1. Figure 7.4 shows an input matrix of size 13 x 12, which contains two exit locations, two evacuees, and two fire nodes. This figure shows the state of fire and evacuees

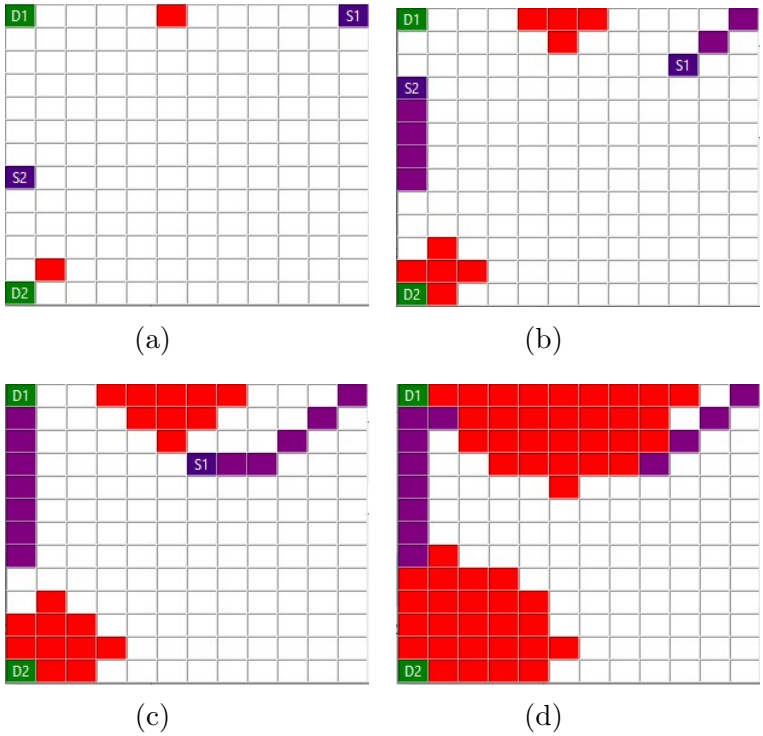


Figure 7.4: A sensor network with 156 nodes, two exits (the green box), 2 evacuees (the purple box) and hazardous area (coloured in red) at (a) t=0 sec (b) t=4 sec (c) t=7 sec (d) t=14 sec.

at 0, 4, 7, and 14 seconds after the detection of fire and execution of the DESSN algorithm. In this scenario, both evacuees are guided towards the first exit since the fire is close to the second exit.

2. Figure 7.5 depicts an input matrix of size 15 x 15, which contains one exit location, one evacuee, and two fire nodes. This figure shows the state of fire and evacuee at 0, 4, 11, and 22 seconds after the detection of fire and execution of the DESSN algorithm. In this scenario, there is only one door, and both fire nodes are between the evacuee and the door. The proposed DESSN algorithm evacuates the evacuee through a safe path. The algorithm designs a long path away from the fire instead of taking a short path that is close to the fire.

3. Figure 7.6 illustrates an input matrix of size 10 x 10, which contains two exit

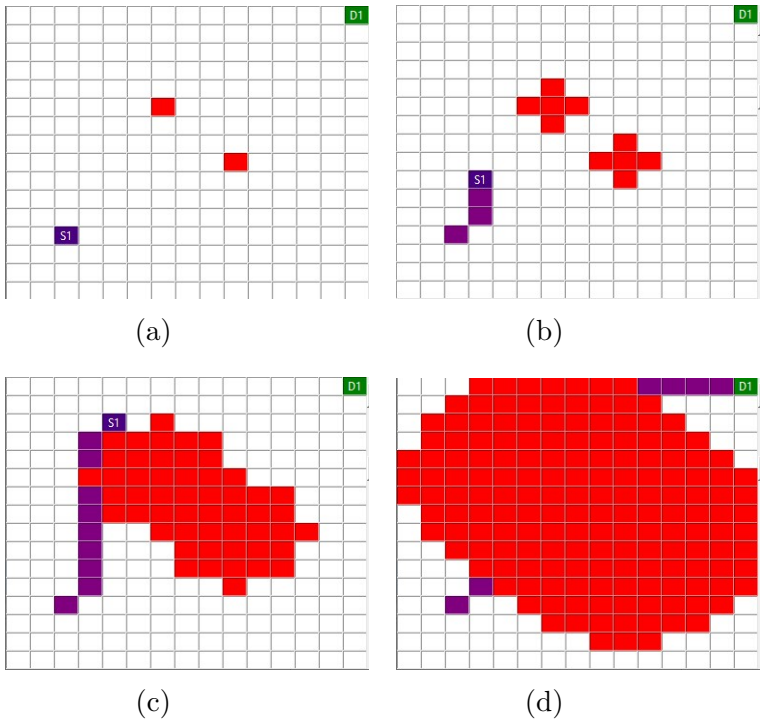


Figure 7.5: A sensor network with 225 nodes, one exit (the green box), one evacuee (the purple box) and hazardous area (coloured in red) at (a) t=0 sec (b) t=4 sec (c) t=11 sec (d) t=22 sec.

locations, two evacuees, and three fire nodes. This figure shows the state of fire and evacuees at 0, 2, 7, and 12 seconds after the detection of fire and execution of the DESSN algorithm. In this scenario, the first evacuee is guided towards the closest exit, but the second evacuee is guided towards the exit that is far from him. This is because the fire may block the closest exit by the time the second evacuee reaches there. Thus DESSN designs a path by considering future fire spread

4. Figure 7.7 shows an input matrix of size 25 x 26, which contains five exit locations, eight evacuees, and 13 fire nodes. This figure shows the state of fire and evacuees at 0, 3, 7, and 18 seconds after the detection of fire and execution of the DESSN algorithm. In this scenario, there are no straight paths to exits. The DESSN algorithm finds the shortest safe path for evacuation of evacuees.

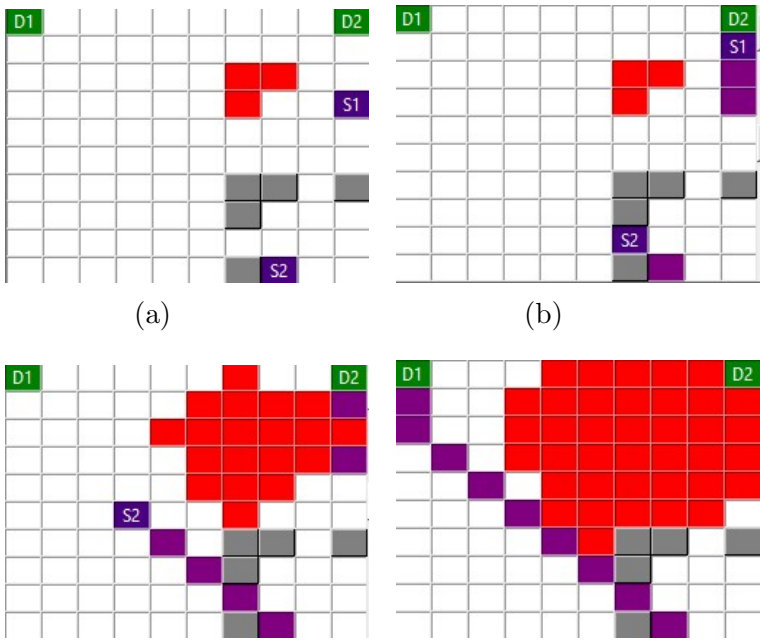


Figure 7.6: A sensor network with 100 nodes, two exits (the green box), two evacuees (the purple box) and hazardous area (coloured in red) at (a) t=0 sec (b) t=2 sec (c) t=7 sec (d) t=12 sec.

Table 7.2: Experimental parameters

Parameter	**Value**
No. of evacuees	500
Evacuees average speed	1.34 m/s
No. of experimental scenarios	5
No. of runs per experimental scenario	10
Run Time (T)	300 sec
No. of exits (Scenario 1)	4
No. of exits (Scenario 2)	3

5. Figure 7.8 shows an input matrix of size 30 x 25, which contains three exit locations, 11 evacuees, and nine fire nodes. The figure shows the floor plan of a library where the state of fire and evacuees at 0, 3, 10, and 21 seconds after the detection of fire and execution of the DESSN algorithm. In this scenario, evacuees are at different locations and there are multiple cyclic paths. The proposed DESSN algorithm finds the shortest safe path to evacuate all the evacuees.

The proposed Dynamic emergency Evacuation for Shortest-Safe path Navigation (DESSN) algorithm is compared with the state-of-the-art emergency evacuation algo-

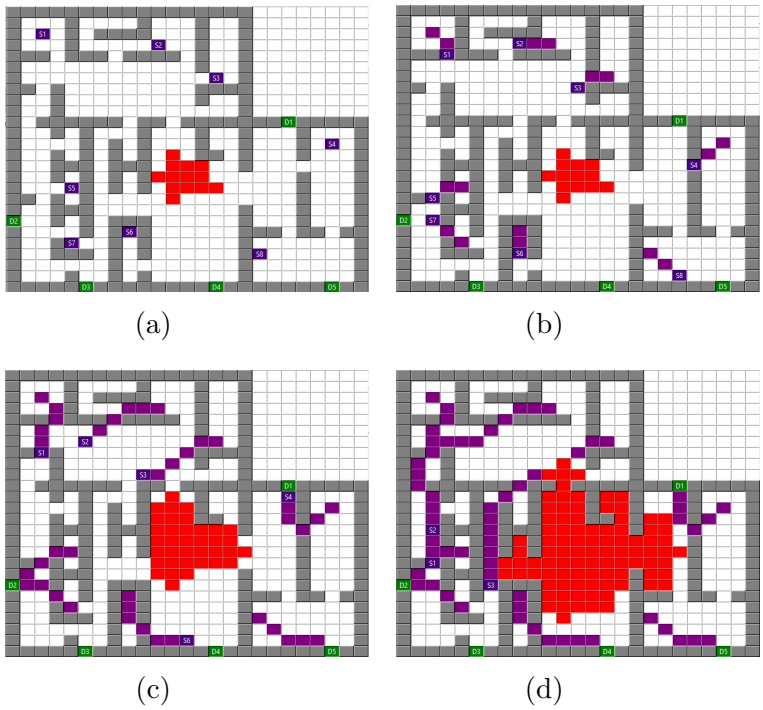


Figure 7.7: A sensor network with 650 nodes, five exits (the green box), eight evacuees (the purple box) and hazardous area (coloured in red) at (a) t=0 sec (b) t=3 sec (c) t=7 sec (d) t=18 sec.

rithms such as (ECSSN) [88], (TODSU) [43] and (FEBIM)[42]. As shown in Figure 7.9a, a map of an equipment and machinery manufacturing plant is used for running the experimental scenarios. This map consists of four exit sites located in four diverse directions. The green region represents the exit sites. The black region represents encompassed areas like machinery and manufacturing units. These black regions act as obstacles that evacuees can't cross. Evacuees can easily traverse the white region. Figure 7.10a shows the floor plan of a public library with three exits. The black regions represent the walls and large bookshelves that act as obstacles for evacuees during emergencies. Experimental parameters are displayed in Table 7.2. 500 evacuees are placed randomly in the white region in each experimental scenario. Each evacuee's average speed is taken as 1.34 m/s. All the evacuees are alerted simultaneously by the fire alarm when a fire emergency occurs.

Five fire-initiating points are shown in Figure 7.9b and Figure 7.10b, which are taken

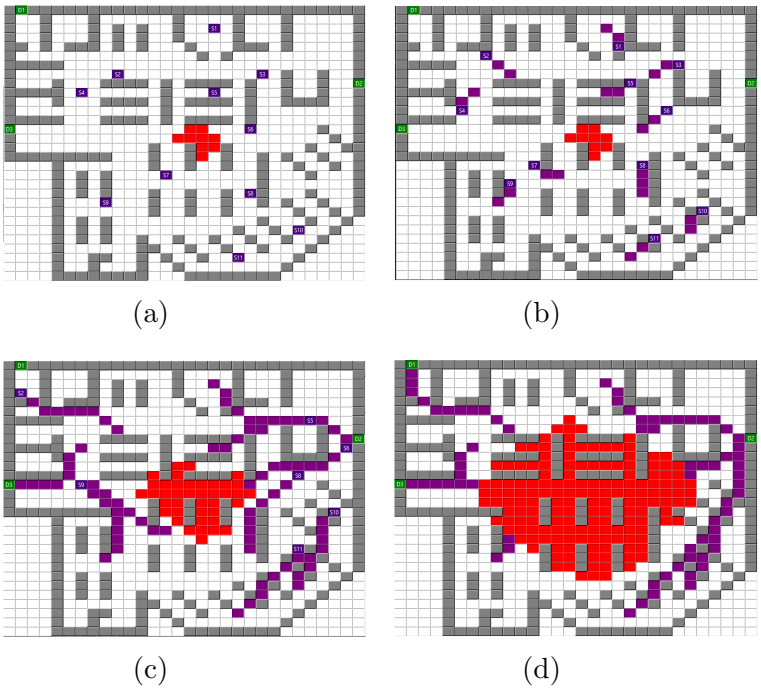
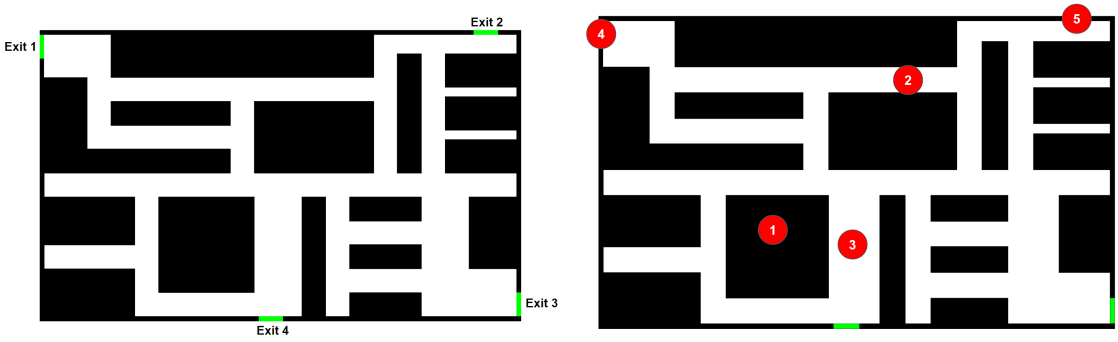


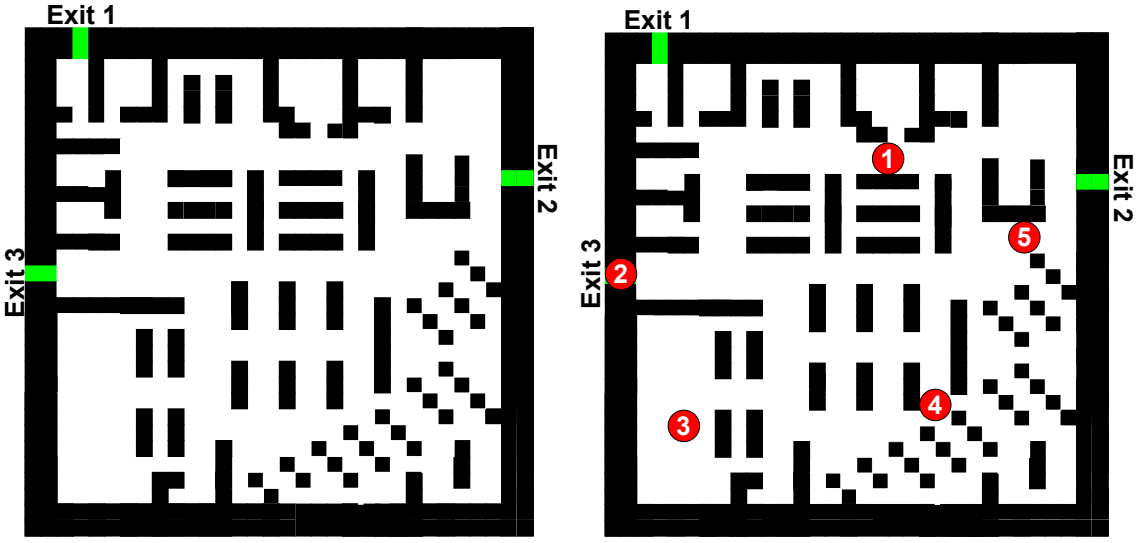
Figure 7.8: A sensor network with 750 nodes, three exits (the green box), 11 evacuees (the purple box) and hazardous area (coloured in red) at (a) t=0 sec (b) t=3 sec (c) t=10 sec (d) t=21 sec.

to minimize the influence of some extra variables. Each initiating point has a distinct fire initiation location. In the first scenario of the first site map, fire takes place in enclosed areas of warehouses and manufacturing workshops, which does not impact the smoothness of the open paths. In scenarios two and three, fire initiations are in open paths. In scenarios four and five, exits are completely blocked after fire initiation occurs. In scenarios one, three, and four of the second site map, a fire occurs in open paths. In scenario two, one of the exits is completely blocked by fire. In scenario five, a fire occurs near an exit. The experiment is executed in each scenario ten times. The average of total evacuation time (time taken for evacuating all the evacuees) from all ten runs on a particular scenario is taken as the result for the comparison of different algorithms. Figure 7.11a shows the comparison of evacuation time for the first site map. Figure 7.11b shows the comparison of evacuation time for the second site map. In both site maps, the total evacuation time is less when the fire initiation point is in a closed



(a) Experimental site map 1. (b) Fire locations on site map 1.

Figure 7.9: Experimental site map 1 and fire locations.

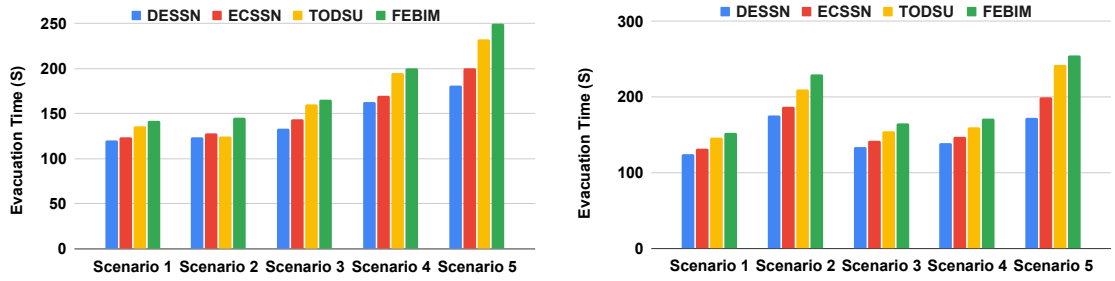


(a) Experimental site map 2 (b) Fire locations on site map 2

Figure 7.10: Experimental site map 2 and fire locations.

space or away from the exit. If the fire source is away and all exits are accessible, then individuals can escape faster. If one or more exits are blocked, then evacuation time is more. This is because individuals near the fire region need to take a longer route to reach other exits. The congestion at the exit also increases due to the unavailability of the other exit. The average evacuation time of DESSN is 25.84% better than ECSSN, 37.87% better than TODSU, and 45.11% better than FEBIM.

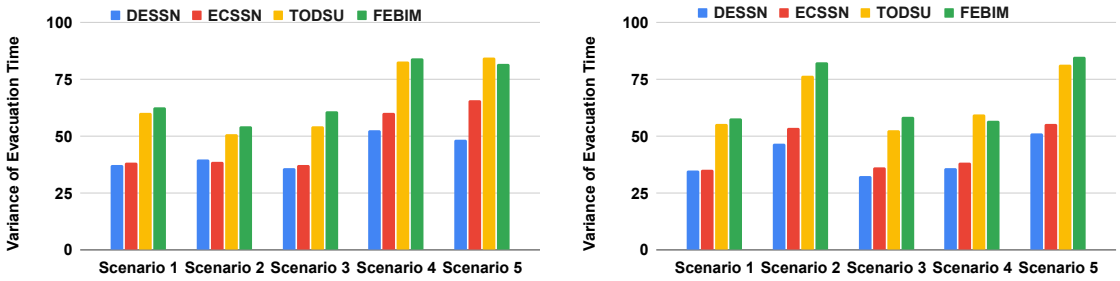
Figure 7.12a and Figure 7.12b show the variance of evacuation time of different



(a) Evacuation time for site map 1.

(b) Evacuation time for site map 2.

Figure 7.11: Comparison of evacuation time.



(a) Variance of evacuation time for site map 1.

(b) Variance of evacuation time for site map 2.

Figure 7.12: Comparison of variance of evacuation time.

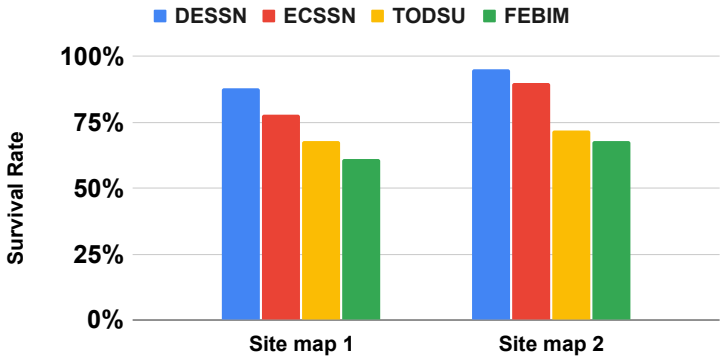


Figure 7.13: Survival rate.

algorithms. Obtaining low variance is better as it shows that evacuation time is closer to the average value. The results show that the DESSN achieves the least variance in all scenarios. Figure 7.13 shows the survival rate for both site maps in case fire breaks

out at points three and five simultaneously. The survival rate is calculated by the ratio of number of successful simulation executions to the total number of simulation executions. A simulation is considered successful if all of the evacuees reach the exit without being affected by the fire. The result shows that the DESSN approach has the highest survival rate. This is because the DESSN approach not only considers the current fire situation but also considers the spread of fire with time while designing the evacuation path. It enables quick evacuation and increases the survival rate. As per the experimental results, it can be concluded that DESSN performs best among other state-of-the-art algorithms because it considers the dynamic spreading of fire to calculate the shortest and safest path.

7.5 Summary

This chapter presents an indoor fire emergency evacuation system using IoT-enabled WSNs for smart buildings. The proposed algorithm efficiently designs optimal evacuation routes for trapped evacuees. The proposed scheme ensures a safe evacuation in both present and future fire situations. Therefore, the proposed DESSN algorithm effectively minimizes large detours, backtracking, panic, and casualties. Simulation results illustrate that the proposed DESSN algorithm outperforms state-of-the-art algorithms. The proposed DESSN algorithm efficiently identifies the shortest and safest evacuation paths by considering current and future fire scenarios.