Volume 38 (1), pp. 1–28
http://orion.journals.ac.za

ORiON
ISSN 0259–191X (print)
ISSN 2224–0004 (online)
©2022

A grid-based humanitarian logistics solution

T Olivier* H Kruger†

Received: 13 April 2022; Revised: 5 May 2022; Accepted: 5 May 2022

Abstract

Natural disasters often cause large-scale destruction and many studies focus on the support and evacuation of disaster victims. During a disaster, timely provision of relief items, such as medical supplies, is a critical task and is considered one of the fundamental functions in a humanitarian logistics chain. However, many practical problems and challenges often occur that make the smooth operation of humanitarian logistic functions a difficult and sometimes impossible task. One such problem is the accessibility to disaster areas where roads, railway lines and other transport routes have been demolished and which causes residents or other victims to be cut off from any assistance. In the event that no or little infrastructure exists, humanitarian workers must find new and innovative ways of reaching people in need. In this paper, a grid-based maze that can be solved to find optimal traversable routes in a disaster area is proposed. A matrix maze generation approach is suggested that can be solved by the Lee algorithm to find an optimal route. To illustrate the proposed methodology, a software solution was developed and applied to a real-world case study. The results obtained confirm that the proposed methodology, combined with the Lee algorithm as a solution strategy, delivers useful and accurate results that humanitarian workers may utilise to assist with the evacuation of victims and the transportation of relief items.

Key words: Grid-based maze, Humanitarian logistics, Lee algorithm, Natural disaster, Shortest route.

1 Introduction

The World Health Organisation defines a disaster as "an occurrence disrupting the normal conditions of existence and causing a level of suffering that exceeds the capacity of adjustment of the affected community" [62]. A natural disaster refers to an event in nature (*e.g.* droughts, earthquakes, hurricanes, floods, tornados) that results in

*School of Computer Science and Information Systems, North-West University, Potchefstroom Campus, South Africa, email: thean0723@gmail.com

†School of Computer Science and Information Systems, North-West University, Potchefstroom Campus, South Africa, email: Hennie.Kruger@nwu.ac.za

http://dx.doi.org/10.5784/38-1-1

fatalities, property damage, and social environment disruption [64]. According to Shaluf [54], natural disasters are typically classified into five broad categories which include geophysical (earthquakes, landslides, tsunamis, volcanic activity), hydrological (avalanches and floods), climatological (extreme temperatures, droughts, wildfires), meteorological (cyclones, storms, wave surges) and biological (disease epidemics, insect/animal plagues). The average death rate of people killed each year globally in these natural disasters is 60 000, with droughts, floods and earthquakes identified as the deadliest current events [52]. From these brief remarks, it is clear that natural disasters are a worldwide phenomenon that can cause great destruction and loss of lives. Many examples of natural disasters that caused significant disruption exist – *e.g.* the November 2019 wildfires that spread across the South Wales region of Australia, where thousands of people had to be evacuated and at least 3000 houses completely destroyed or seriously damaged [12]; Typhoon Hagibis which caused mayhem in early 2019 in Japan resulting in the evacuation of about 38000 people with more than 138000 households without water and almost 500000 without electricity [46]; and in the United States of America, Hurricane Sandy caused the death of approximately 220 people in 2012, destroyed more than 600000 housing units and caused almost $70 billion in damages [21]. More recently, Hurricane Ida cost the world $65 billion in losses after hitting the state of Louisiana, USA, in August 2021 [33], while a storm complex over Europe caused the loss of lives, buildings to collapse, and destroyed roads and railways from 12 to 15 July 2021 [60].

In the aftermath of a natural disaster, one of the crucial tasks is to make provision for relief and medical teams to assist victims. Medical, food and relief items (*e.g.* shelter, clothing, personal hygiene, *etc.*) need to be provided to disaster victims in a timely manner. This critical task is managed through a process called a humanitarian logistics chain which may be defined as the planning and implementation of efficient and cost-effective procedures to manage and control the flow and storage of goods and relief items between a point of origin and people affected by the disaster [18]. However, following a natural disaster, the smooth operation of the so-called humanitarian logistics chain is often confronted with a significant number of challenges and practical difficulties. One of the typical problems associated with natural disasters is the accessibility of certain areas to reach disaster victims. Xu *et al.* [64] pointed out that roads, communication and power sources are often damaged by severe natural disasters, while Bil *et al.* [9] confirmed that roads and transport infrastructure is often demolished, causing residents and other disaster victims to be cut off from the outside world and from any help. To find an optimal traversable route in an area with no or little infrastructure, that can be used to transport relief resources to victims, lies at the core of humanitarian logistics. It often leads to new and innovative ways of reaching people in need and evacuating them to appropriate and safe locations.

In this paper, a grid-based maze approach that can be solved with the well-known Lee algorithm [38] is proposed to assist with the primary humanitarian logistics problem of finding a suitable route, or routes, in a disaster-stricken area. A grid-based maze is constructed for a real-world natural disaster case. The maze is then evaluated, using the Lee algorithm to solve the primary humanitarian logistics problem of finding an optimum traversable route between different pre-specified locations. Real-world data from Hurricane Katrina, which hit New Orleans in the United States of America in 2005, will be used to demonstrate the proposed solution strategy.

The remainder of the paper is structured as follows. A brief overview of related work in the literature is presented in Section 2. The generation of a grid-based maze is explained in Section 3 while an overview of the Lee algorithm as a solution strategy is provided in Section 4. A real-world case study is illustrated in Section 5. Section 6 presents a discussion of the results, and the paper is then concluded in Section 7 with opportunities for further research as well as some concluding remarks.

2 Related work

Natural disasters and the associated large-scale destruction caused by them are an important subject of study. Many researchers performed studies in this area with topics that range from understanding the nature of natural disasters to the support and evacuation of disaster victims, *e.g.* humanitarian logistics. Recent examples of such studies include [2], [10], [42], and [44]. There is also an abundance of studies that focus exclusively on humanitarian logistics. For example [23] proposed a decision support system to address critical time management in humanitarian logistics, while Shao *et al.* [55] reviewed deprivation cost (qualification of human suffering) in humanitarian logistics. Lukosch and Comes [41] implement gaming to simulate specific environments and conditions in humanitarian logistics, whereas Sigala and Wakolbinger [57] proposed outsourcing to appropriate specialist service providers as a viable option in humanitarian logistics. Further examples of research studies that show the dynamic nature of humanitarian logistics can be found in [30], [53], and [63].

A grid map, defined as a graphical representation of a region that is divided into equal sub-sections where each sub-section represents a specific measured distance [25], is typically used where a form of navigation between locations on a map is required. Grid maps form the basis for many scientific studies and are widely acknowledged as a useful method in pathfinding and other studies. Examples include Bekker and Schmid [8], who used a grid map to determine safe routes for ships travelling through sea minefields; Nelson and Smith [45] implemented a grid map in gaming where objects are placed on a grid as obstacles to certain travelling routes; grid maps can also be used to represent populations over large geographical areas [7], or as mapping frameworks for mobile robots [19]. Additional applications of grid maps are detailed in [34] and [58].

Lee *et al.* [39] defined a maze as a grid consisting of cells with an entrance (starting cell) and an exit (ending cell) together with walls that ultimately form both traversable paths as well as dead ends. There exist several algorithms that can be used to generate a maze - *i.e.*, algorithms based on greedy principles [20], spanning tree approaches [35], and graph theory [11]. Arguably the most popular algorithms are Prim's algorithm [49] and Kruskal's algorithm [36]. Mazes prove to be standard tools in many research projects and are frequently used as a technique in studies on various scientific subjects. For example, Pershin and Di Ventra [47] employed mazes as a tool to test the proficiency of memristor networks (resistors within memory), while Adamatzky [1] studied the behaviour of different organisms with the help of maze structures. Other examples may be found in [3] and [40].

Grid-based maze structures can be constructed by using a grid map combined with maze generation algorithms (a grid-based maze structure will be discussed in Section 3).

These structures allow for the representation of real-world environments in a maze where real-world obstacles are represented by walls in the maze [32]. As with maze generation algorithms, there also exist a wide variety of techniques and algorithms that can be used to solve grid-based maze structures – some of the standard algorithms include the Lee algorithm [38], which will be elaborated on in Section 4, the A-star algorithm [26], the flood-fill algorithm [37] and the recursive backtracking algorithm [61]. The many uses of grid-based maze structures are visible in research studies such as the study by Jannu and Jana [31], who utilised a grid-based structure in the planning and implementation of a clustering and routing algorithm to solve problems in wireless sensor networks. Barnouti *et al.* [6] combined the use of a grid and maze routing algorithm to find shortest paths in gaming. Other examples of studies relevant to grid-based maze applications can be found in [22] and [45].

To conclude this section, a brief mention is made of the large number of studies in mathematical and computational modelling techniques that are frequently used in humanitarian logistics. Shortest path models are of particular interest in this study, and in humanitarian logistics, and many research papers exist on this topic. Probably the most well-known shortest path algorithm was formulated by Dijkstra [16] and is based on an iterative modification process of labels. A label details the immediate predecessor of a node (in a network) on the current shortest path as well as the length of the current shortest path. Labels and shortest path predecessors are iteratively revised and reduced until the shortest route is identified. Examples of studies on Dijkstra's algorithm related to natural disasters can be found in [27] and [43]. Another popular algorithm that may be employed to find the shortest route under specific circumstances is the A-star algorithm [26]. The algorithm, which is defined as a best-first algorithm, uses a combination of heuristic search and shortest path searching techniques to identify an optimal route [17]. The work of Chaudhari *et al.* [13] serves as an example of applying the A-star algorithm to find an optimal path in a maze. The Lee algorithm is proposed in this paper as a solution strategy to find an optimal path in a grid-based maze structure and will be discussed in detail in Section 4. Examples in the literature of applications of the Lee algorithm can be found in Polanczyk *et al.* [48], who proposed the use of the Lee algorithm (as opposed to the A-star algorithm) in scenarios with dynamic environments where locations and obstacles may change; Daneshjo *et al.* [14] implemented the Lee algorithm in a robot environment to ensure that robots can move freely and avoid collisions with obstacles; and Reddy *et al.* [51] utilised Lee's algorithm to address various routing constraints.

Humanitarian logistics is a far-reaching area of study and does not only refer to the formulation of shortest path models and mathematical techniques to determine an optimal route. A wide range of other studies, particularly the development of mathematical models, are frequently undertaken in humanitarian logistics. Such studies range from facility location models [56] and vehicle routing optimisation [5] to humanitarian logistics network design [50]. For a systematic review of contemporary literature on humanitarian logistics [29] may be consulted.

3 Generating a grid-based maze

In this paper, a grid-based maze structure (solved by the Lee algorithm) is proposed to assist in humanitarian logistics in the event of a disaster. The relevant concepts were briefly defined in the previous section, and the aim of this section is to introduce the idea of generating a grid-based maze structure that may be used for further analysis. The Lee algorithm as a solution methodology will then be explained in Section 4.

Once a grid has been constructed, a maze can be generated in such a way that the edges of each cell within the grid represent a possible wall in the maze. Figure 1 shows an empty 6×6 grid on the left and a maze generated on the grid on the right-hand side. The maze walls (that prevent movement) are indicated in thicker bold black lines. The traversal between cells can only occur in horizontal and vertical directions, and no diagonal movements are allowed.

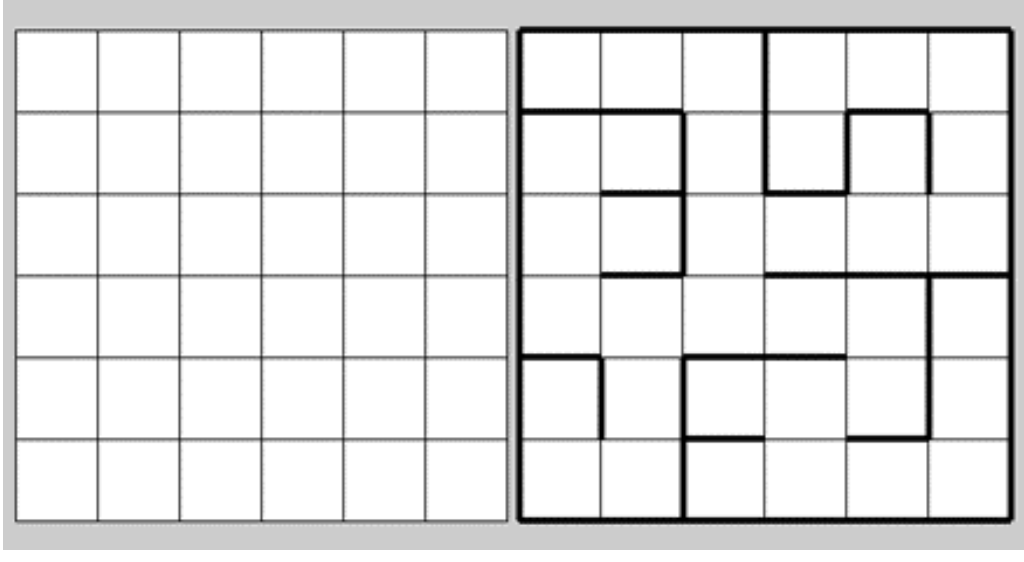

Figure 1: A simple 6×6 grid-based maze

The use of such a grid-based maze is of significance because it allows for real-world scenarios and geographical areas to be represented as a maze [32]. It is particularly useful in humanitarian logistics as a disaster-stricken terrain can be described as a grid-based maze where walls are indicated as inaccessible areas in the real-world disaster-stricken area. If the grid-based maze is then solved (finding an optimal path between specific location points), an equivalent optimal path is found simultaneously in the real-world scenario, thus allowing for a more efficient humanitarian logistics activity through the predetermining of an optimal path to traverse by rescue workers. To illustrate these practical advantages, consider Figure 2, which depicts large-scale destruction and damage in North Carolina (USA) caused by Hurricane Florence in September 2018.

Placing a grid over the disaster area, a maze can be generated by blacking out those cells that are not reachable via any form of transportation such as a boat or other vehicle. Air transport is not considered as the terrain does not prohibit the use of helicopters or drones for example. A grid-based maze of the disaster area is displayed in Figure 3. The red line

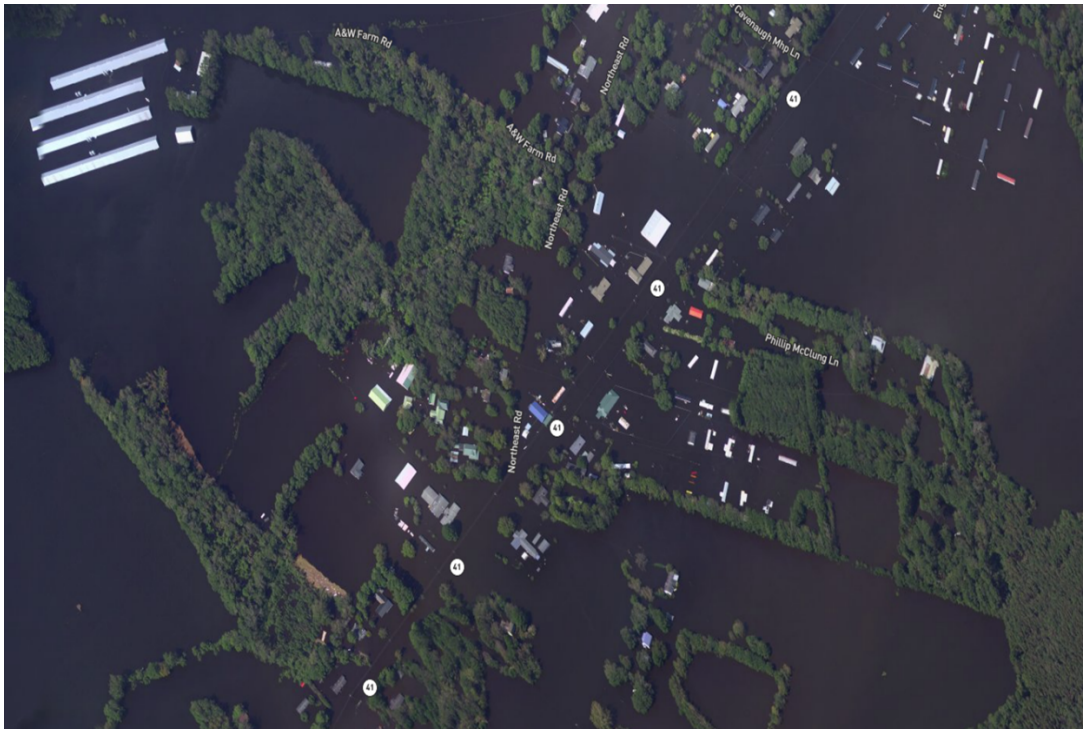

Figure 2: Flooding caused by Hurricane Florence in the town of Cartersville, North Carolina in 2018 (Source: Google Maps Satellite Image)

in Figure 3 is a hypothetical example of a route through the maze.

Figure 4 shows how an equivalent path, indicated again by the red line, to the one in the maze (Figure 3) is constructed simultaneously in the real-world scenario. Rescue workers may now use this optimal path to assist victims.

Several techniques and algorithms may be employed to generate a maze that can be used in scenarios like the one explained above. Reference was made to these algorithms and the approaches that they are based on in Section 2. The standard maze generation algorithms (*e.g.* Prim's and Kruskal's algorithms) use some form of randomisation to construct a maze. This makes the use of traditional algorithms problematic when dealing with a disaster terrain where certain areas cannot be subjected to random placement of walls or paths due to the destruction of areas. To address this problem, a matrix maze generation approach is employed in this study. The proposed method uses a grid, together with the available image or map data of damaged and non-damaged areas in the disaster terrain. Based on the cells of the grid and the associated image data, a matrix (representing the area) is derived where the matrix entries indicate the placement of walls in a maze. This approach is an adaptation of the work of Aki and Gullu [4] who successfully implemented a maze using an image with an associated matrix representation. Details of the implementation of the adapted method used in this study will be presented in Section 5, where a real-world case study is discussed.

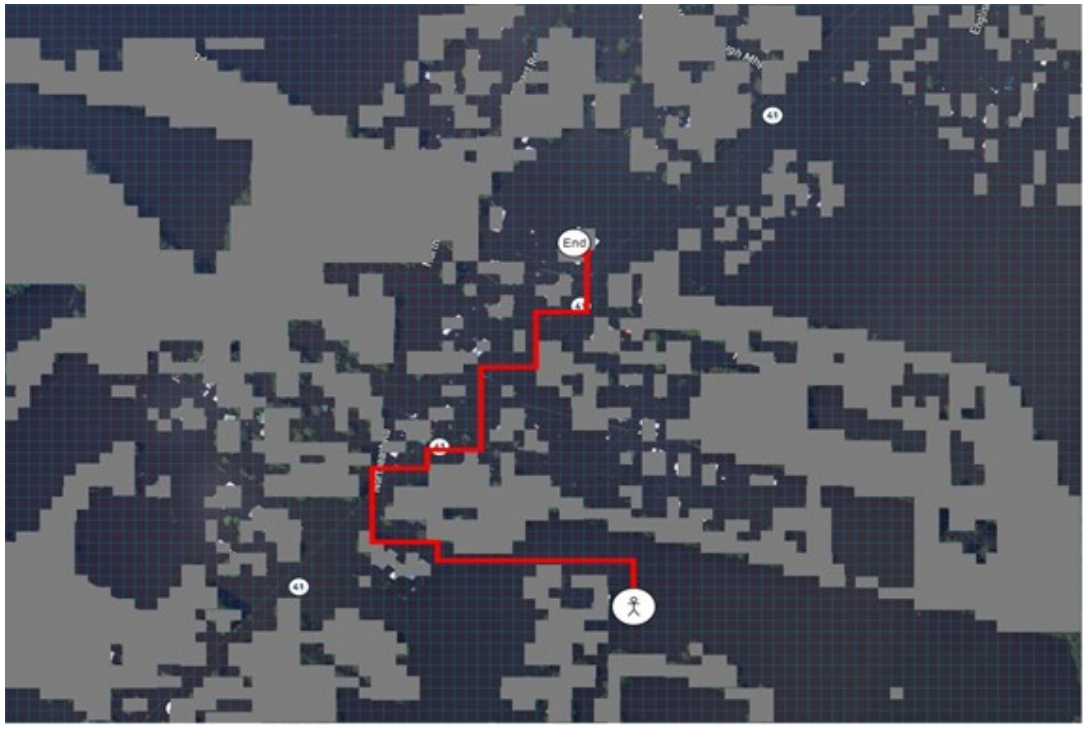

Figure 3: Grid-based maze overlay – Cartersville, North Carolina

Figure 4: Grid-based maze overlay indicating real-world route – Cartersville, North Carolina (Source: Google Maps Satellite Image)

4 The Lee algorithm as a solution strategy

Following the construction of a grid-based maze as described in Section 3, the next step would be to solve the maze. As with the generation of a maze, many strategies, heuristics, and algorithms exist to find a solution. Not all techniques guarantee a solution, while others may only find sub-optimal solutions. An example of a basic technique to solve a maze is the wall follower method. The method is also referred to as the left-hand rule or right-hand rule and operates by simply keeping the left hand, for example, in contact with the wall of the maze while walking through the maze. By doing this, one is guaranteed to find the exit if it exists [28]. There are also more formal algorithms that may be used; these algorithms were highlighted in Section 2. In this study, the Lee algorithm [38] was chosen as a solution strategy as it is one of the algorithms that will always find a solution if one exists.

The Lee algorithm was formulated in 1961. The aim was to find a procedure that would solve path-connection problems efficiently as well as to find optimal routes in route-finding problems. The algorithm is well suited to solve a maze as it can solve related problems such as finding a path between two points while avoiding obstacles (walls in a maze) and finding an optimal path between two points, *i.e.*, the shortest route. Gupta and Sehgal [24] describes the Lee algorithm as a two-stage method. During the first stage, called the *filling* stage, each cell is visited until the goal is reached. In the second stage, called the *retrace* stage, the path to the goal is traced back to the starting point to construct a final path. Using a simple grid, the algorithm can be explained as follows (see Figure 5).

- Start with an empty grid with a starting cell (S) and a goal cell (G) - see Step 1 in Figure 5.
- Enter a 1 in the starting location and a 2 in all the adjacent cells – see Step 2 in Figure 5. These values represent the distance from the starting location.
- Repeat the above step by adding an increased value to neighbouring cells until the goal cell (G) is reached – see Steps 3 and 4 in Figure 5. Note that the goal cell is reached with a value of 7. This indicates the completion of the filling stage of the algorithm.
- The retrace stage starts by backtracking from the goal to the starting location. This is done by selecting an adjacent cell to the current goal cell with a value of i-1, where i represents the value of the current cell – see Step 5 in Figure 5.
- The final path is now generated, resulting in the shortest path between the starting location (S) and the goal (G) – see Step 6 in Figure 5. This example uses an empty grid with no obstacles (walls), and alternative solutions may exist.

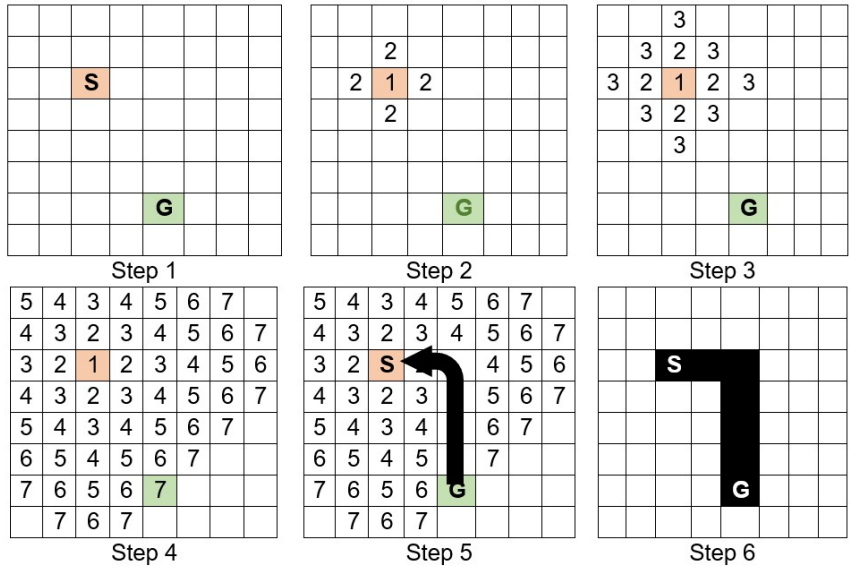

Figure 5: The filling and retrace stages of the Lee algorithm

The illustration in Figure 5 uses an empty grid. However, a grid is only the underlying element of a maze which consists of limitations such as walls that restrict movement. To illustrate how the Lee algorithm solves a maze, the following steps may be used (see Figure 6).

- Given a maze, a valid start cell - indicated in red in Step 1 in Figure 6, and a valid goal cell – shown in green in Step 1 in Figure 6, are selected. A valid cell is a cell that is not a wall, and that has at least one neighbouring cell that can be explored.
- Start at the starting cell and select direct neighbouring cells to be explored – see Step 2 in Figure 6.
- Explore neighbouring cells. Only one neighbouring cell is expanded with each iteration. Exploration is based on a first-in-first-out basis, meaning that oldest expanded cells will be explored first before moving on to respective neighbouring cells – see Steps 3 and 4 in Figure 6.
- Continue to expand each cell iteratively until the goal cell is reached. This signals the end of the filling stage – see Step 5 in Figure 6.
- Start backtracking from the goal cell by adding the goal cell's parent cell to the path. The parent cell is then set as the new current cell and the process is repeated until the starting cell is reached. The path generated with this process will contain the least number of expanded cells and constitutes the shortest path – see Step 6 in Figure 6.

Algorithm 1 presents the pseudo-code of the Lee algorithm as implemented in this study. The algorithm has certain shortcomings (*i.e.*, being relatively slow with high memory requirements); however, it remains one of the best options due to its low complexity and guarantees an optimal shortest path solution if one exists [24].

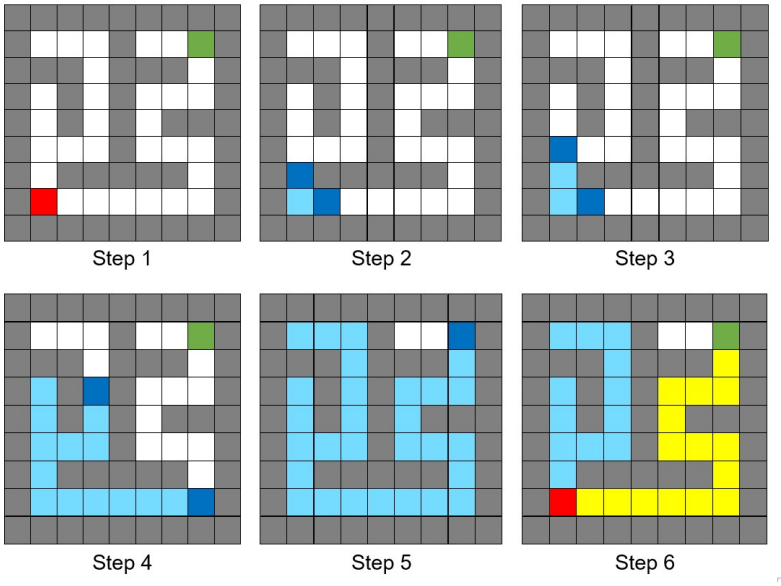

Figure 6: Maze solution generated by the Lee algorithm

Algorithm 1: The Lee Algorithm

```
Input:  start_node          // the source node to start from
        goal_node           // the goal node to reach
Output: PATH - list from start_node to goal_node
lists: OPEN, CLOSE and PATH
1. push start_node to OPEN
2. while OPEN is not empty
3.     current_node ← first element of OPEN
4.     If current_node = goal_node
5.         break to line 15
6.     end if
7.     for each neighbour of current_node
8.         If neighbour is not a wall
9.             push neighbour to OPEN
10.         end if
11.     end for
12.     push current_node to CLOSED
13.     pop current_node from OPEN
14. end while
15. while current_node != start_node          // backtracking phase
16.     push current_node to PATH
17.     current_node ← parent of current_node
18. end while
19. return PATH
```

In the next section, a real-world case study illustrates an implementation of the Lee algorithm.

5 A real-world case study

To illustrate the techniques and methodologies proposed in the preceding sections, a real-world disaster area was chosen, and a computer system was developed to show the practical advantages the suggested methodology may have for a humanitarian logistics operation. The case study section will begin with a short background description of the real-world disaster used in the study, followed by a discussion on how the grid-based maze was developed. A software implementation and the application of the Lee algorithm will then be presented.

5.1 The real-world disaster

Hurricane Katrina, which hit New Orleans in the USA in August 2005, was chosen as a case study. New Orleans (see Figure 7) is a city situated in Louisiana, USA, along the Mississippi River and houses approximately 494 000 residents [15].

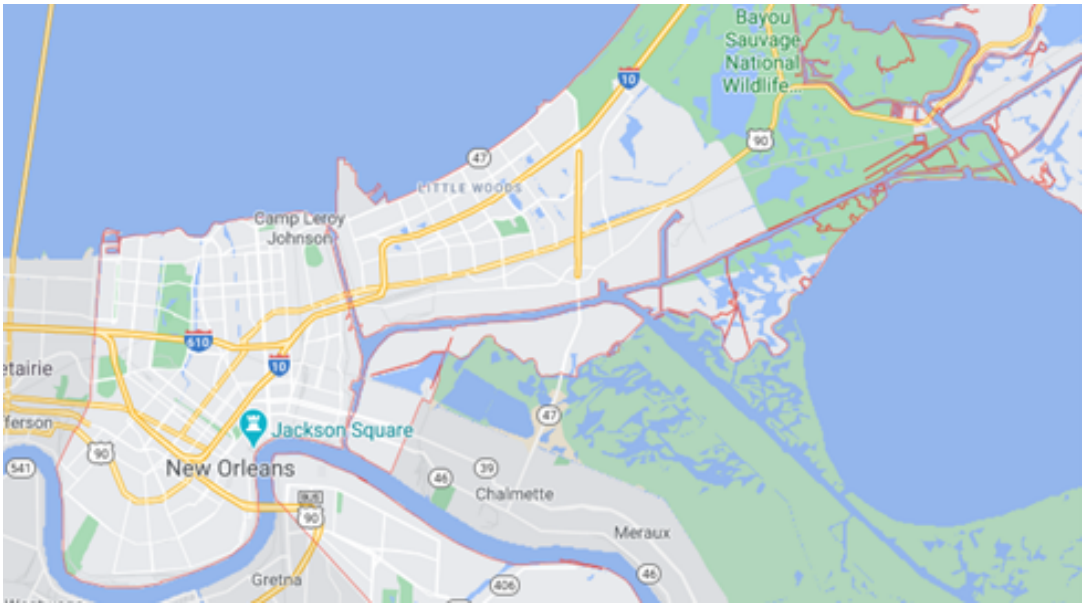

Figure 7: Map of New Orleans (Source: Google Maps)

Hurricane Katrina originated over the Bahamas and hit New Orleans on 29 August 2005. The hurricane was classified as a category five cyclone – the worst type of storm with wind speeds of up to 280 km/h. Following the storm, property damages of $108 million were recorded, while 1388 residents lost their lives [59]. A weather photo of the storm is shown in Figure 8, while Figure 9 shows part of the damage caused by Katrina.

To transform data from a real-world disaster into a grid-based maze, it is necessary to obtain data that accurately depicts the severity of damage caused to various locations in the disaster area. In the case of Hurricane Katrina, detailed data was gathered and interpreted by the *LSU Katrina Survey Team Department of Sociology*[1]. This data

[1]https://www.lsu.edu/faculty/fweil/KatrinaMaps/index.htm

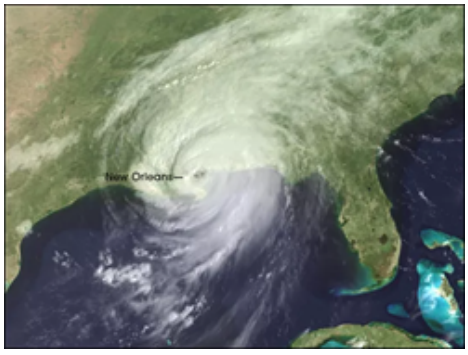

Figure 8: Hurricane Katrina (Source: GOES Project Science Office)

Figure 9: Hurricane Katrina damage (Source: NWS/Lieut. Commander Mark Moran, NOAA Corps, NMAO/AOC)

accurately shows the severity of damage in different areas of New Orleans. The accurately recorded data is of particular interest in this study as it enables the construction of a matrix that can be used to generate a maze of the area. The data is shown in Figure 10, where damage severity is indicated by means of a colour code. Areas where no damage occurred are indicated by green dots; yellow dots are used to show signs of damage, but not complete destruction; and orange and red dots are used to mark areas where major damage occurred. The major damage varies from partially destroyed buildings and roads to completely demolished buildings and roads. In the context of this study, where a maze will be generated to facilitate path selection, the orange and red dots represent inaccessible areas that will be translated to walls in a maze. The green and yellow dots represent accessible areas and will indicate possible paths in a maze.

As depicted in Figure 10, the severity levels of damage can now be used to develop a grid-based maze that the Lee algorithm will ultimately utilise to determine optimal paths in the disaster area.

5.2 Generating a grid-based maze for the Greater New Orleans area

The steps to generate the grid-based maze are summarised as follows.

1. Grid placement over the disaster area;
2. Matrix development to represent the disaster area;
3. Generate a maze from the matrix; and
4. Fit the final maze over the disaster-stricken area under consideration.

Grid placement over the disaster area

A grid size of 70×70 was chosen. There is no exact rule or algorithm to determine a suitable grid size for the disaster area used in the context of this study. The grid-size decision was therefore based on a number of experiments to determine an appropriate size. The main criterion was that the grid must provide adequate detail within each grid

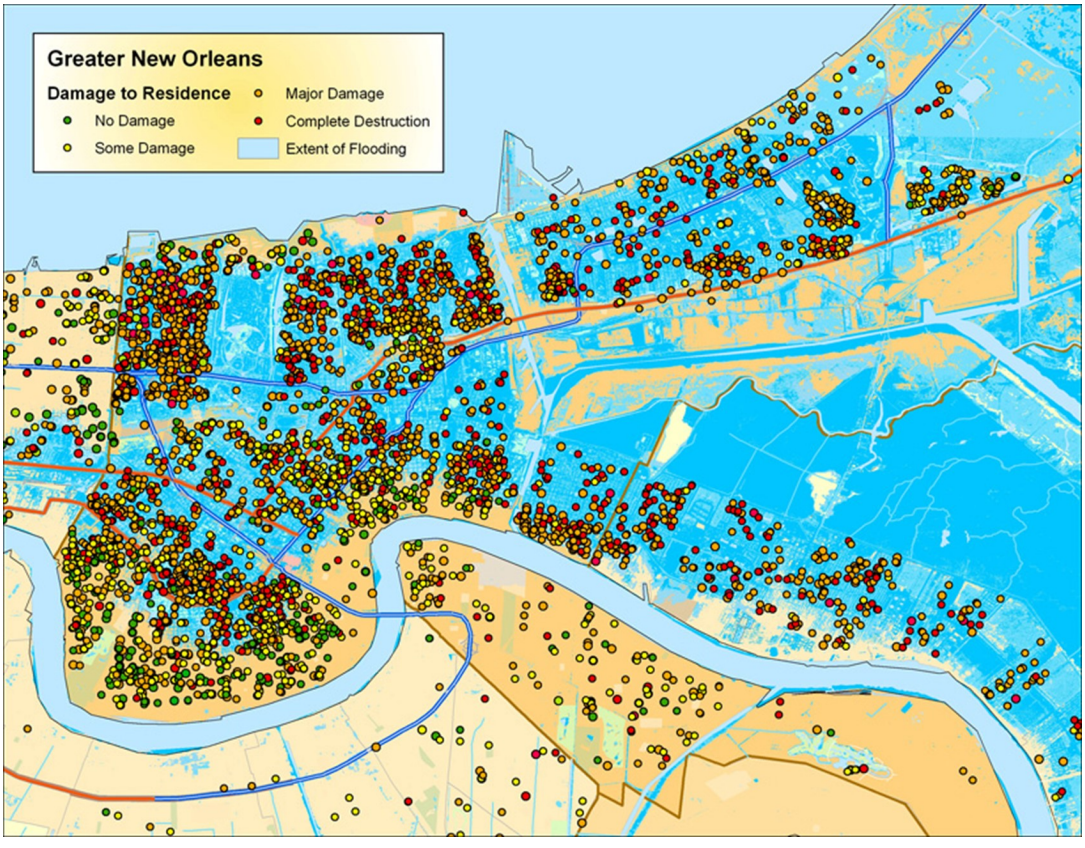

Figure 10: Severity classification of damage to the Greater New Orleans area (https://www.lsu.edu/fweil/KatrinaMaps/index.htm)

cell to construct a representative matrix of the disaster region, which can be translated into a representative and reliable maze. The 70×70 grid was placed over the Greater New Orleans map and is presented in Figure 11. Note that the grid size in Figure 11 has been adjusted for visual and presentation purposes and does not reflect an actual 70×70 grid.

Matrix development to represent the disaster area

The development of a matrix to represent the disaster area is based on an adaptation of the work of Aki and Gulla [4]. A matrix representing the area in Figure 11, can automatically be developed using image recognition software to recognise the different colour-coded destruction points. However, in this study, such specialised software was not considered and the matrix was developed manually based on the damage points and the 70×70 grid. This was a relatively simple task and was performed by assigning different values to different combinations of walls for a cell. For example, an entry of 1 in the matrix indicates the existence of a wall on the left border of the cell. The value assignment was done according to the details in Table 1.

Note that the four values $0 - 3$ are sufficient to cater for all possible combinations of walls, *e.g.* should a wall be required at the bottom border of a cell, a value of 2 (wall at top of

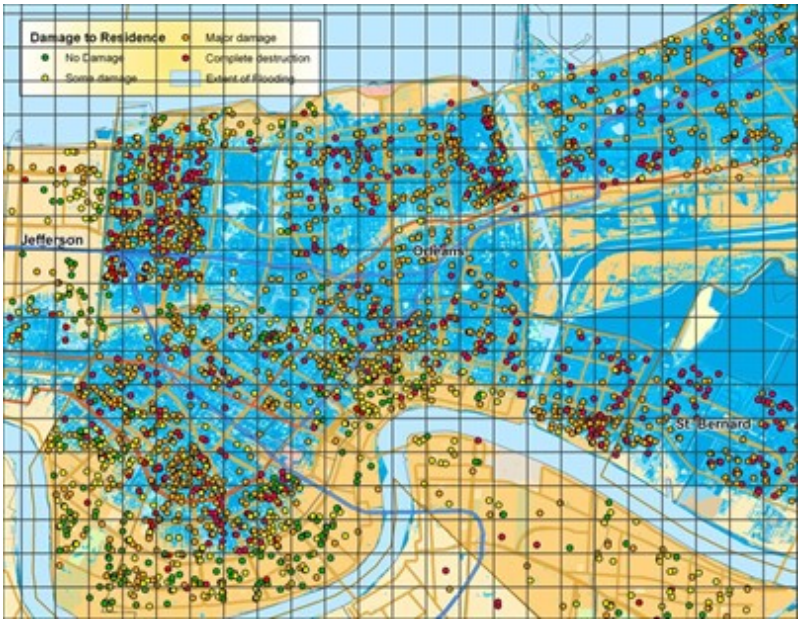

Figure 11: Greater New Orleans disaster area with grid overlay

Table 1: Matrix value assignment.

Description	Graphical representation	
	Matrix value	Wall
No walls in the cell	0	
Wall to the left of the cell	1	
Wall at the top of the cell	2	
Wall to the left and the top of the cell	3	
Inaccessible area	4	

the cell) may be inserted in the cell immediately below the cell that requires the wall at the bottom border. Other combinations can also be addressed by using the existing four options.

The actual assignment of the numbers is shown in Figure 12 where a small extract of the map of the disaster area is shown on the left. This picture is then visually inspected to derive the matrix values in the middle of Figure 12. Finally, using the value assignment rules in Table 1, the walls are constructed (on the right of Figure 12) using the matrix entries.

The complete 70×70 maze generating matrix is presented in the Appendix.

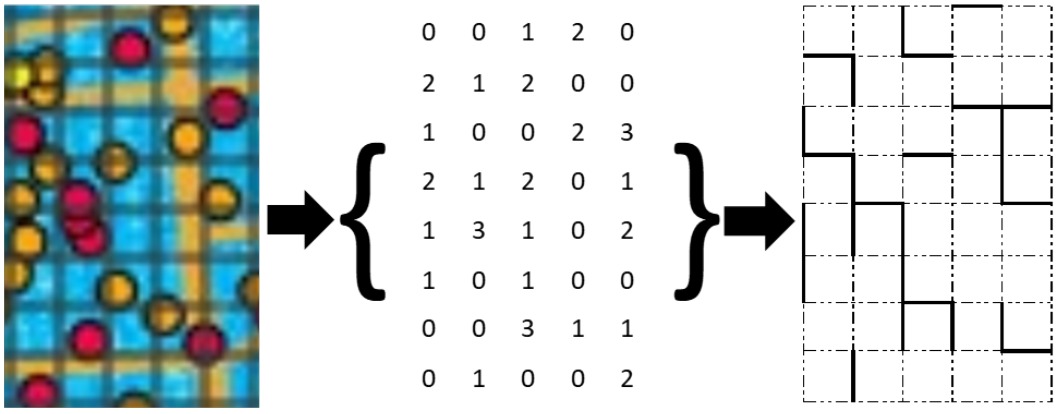

Figure 12: Example of the maze generation process using a matrix

Generating a maze from the matrix

The 70×70 matrix that was constructed according to the rules presented in the foregoing paragraph, was used to generate a maze structure that represents the entire disaster region. Software to perform the maze generation was developed using Visual Studio and the C# programming language. Algorithm 2 details the pseudo-code of the maze generation process using the 70×70 matrix.

Algorithm 2: Matrix maze generation

```
Input:  MAZE            // 2D array of maze matrix
        gridPen         // used to draw the maze walls
        gridBrush       // fills inaccessible areas of the map (e.g. river
                        and sea)
        HospitalBrush   // used to indicate locations of hospitals
Output: g               // graphics of completely generated maze
Process:  Generating a maze using a matrix
1. For each value in MAZE
2.     if value = 1
3.         g.draw left wall using gridPen
4.      else if value = 2
5.         g.draw top wall using gridPen
6.      else if value = 3
7.         g.draw left wall using gridPen
8.         g.draw top wall using gridPen
9.      else if value = 4
10.         g.fill cell with gridBrush              // inaccessible area
11.     else if value = 5
12.         g.fill cell with hospitalBrush          // relief facility
13.     end if
14. end for
```

Fit the final maze over the disaster area

The maze generated for the New Orleans disaster area is displayed in Figure 13. The green and blue squares are illustrative and used as reference points in the final path generating examples. The squares were chosen randomly and represent starting points (green squares) and endpoints (blue squares) for the Lee algorithm to find an optimal route. A starting point may be an area where victims are trapped, while an endpoint may indicate a medical or relief station.

It is important to note that the maze structure in Figure 13 contains a large number of open cells – these cells were not catered for in the matrix wall generation process. The open cells indicate areas that are accessible, and it may be argued that the maze is therefore sufficient. However, there must be a complete maze with no open cells to apply a maze-solving algorithm. To address this problem, any standard maze generation algorithms (*e.g.* Prim's or Kruskal's algorithm) are typically used to complete the maze and eliminate the empty cells.

For this study, it was necessary to choose Kruskal's algorithm to complete the maze. Prim's algorithm was not an option as the algorithm starts with a single randomly selected cell and then explores in different directions to place walls. This makes the use of the algorithm infeasible as the existing walls (generated by the matrix) would block the algorithm's search path and leave some of the cells unexplored. As opposed to this, Kruskal's algorithm chooses a random starting point with each iteration, providing a means of exploring all the cells regardless of the walls placed by the matrix approach. Due to the randomisation at each iteration, Kruskal's algorithm is the ideal candidate to combine with the matrix approach and to ensure that a complete maze is constructed without any unexplored or empty cells. However, the algorithm may still create dead ends between the matrix-generated walls and the walls generated by Kruskal's algorithm. To avoid any dead ends, the algorithm was adapted to keep neighbouring cells (of the matrix generated maze) empty to allow for open paths between the matrix-generated maze and the maze generated by Kruskal's algorithm. The pseudo-code of the adapted Kruskal algorithm is presented in Algorithm 3.

Algorithm 3: Adapted Kruskal maze generation algorithm

```
Input:  S       // Set of a collection of sets containing cells to explore
        X       // Set of cells in Matrix Maze
Output: M       // Maze is generated
Process: Generating a perfect maze from a set of cells by Kruskal's
algorithm
1.  declare e and c1, c2          // edge and cells
2.  Select a random edge e = (c1, c2) ∈ S
3.  M ← (c1, c2)
4.  while number of sets in S>1 // all cells do not belong to the same set
5.      Select random edge e = (c1, c2) ∈ S with c1 and c2 in different sets
6.      If e ∈ X                  // modified statement
7.          M ← M ∪ {empty cell} // ensure no dead-ends occur
8.      else
9.          M ← M ∪ {(c1, c2)}
10.     end if
11.        unify c1 and c2 in S into a single set
12. end while
```

The final and complete maze generated by the matrix approach combined with the adapted Kruskal algorithm is shown in Figure 14.

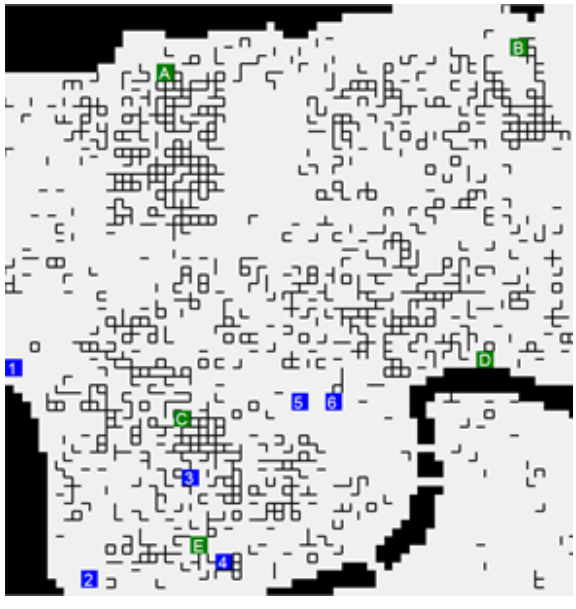

Figure 13: New Orleans maze from matrix

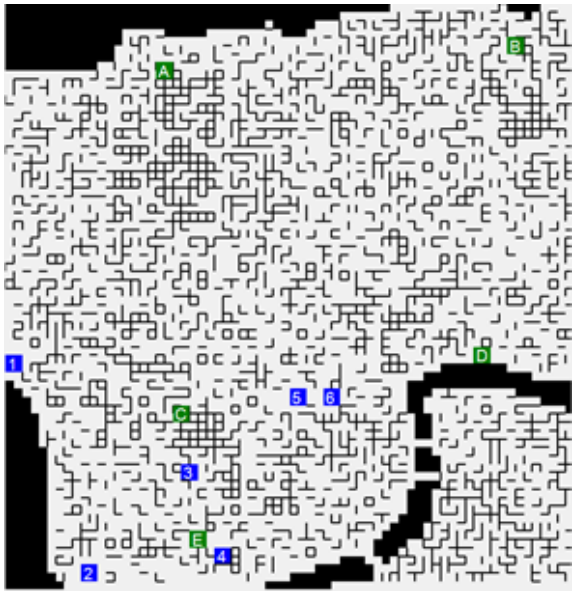

Figure 14: Complete New Orleans maze

5.3 Solving the maze with the Lee algorithm

To illustrate how the grid-based maze developed for the New Orleans disaster area (see Figure 14, Section 5.2) can be solved using the Lee algorithm, a software solution was created and implemented. The software demonstration system was developed using Visual Studio and the C# programming language. The solution strategy is based on the Lee algorithm presented in Algorithm 1 and explained in Section 4.

The system displays the maze and then provides several options to a user. The final optimal route, as determined by the Lee algorithm, is then mapped out on the maze. Figure 15 presents the user interface, the options available to a user, and the final optimal route generated by the Lee algorithm.

As mentioned earlier, the five green blocks represent starting points (*i.e.*, where victims are located), and the six blue blocks indicate the possible endpoints (*i.e.*, relief or medical stations). Note that some of the green starting points are obscured by the blue expansion area generated by the Lee algorithm – see Figure 14 for all the green starting point locations.

The different options available to a user are shown on the left-hand side (in Figure 15). The system also provides for using the A-star algorithm as a solution strategy - this option is not elaborated on in this paper. In this specific example, starting point B was selected as well as all six relief stations (hospitals) indicating that all six of them are available as an end or evacuation point. On the right-hand side, the blue area indicates the filling stage of the Lee algorithm, while the red line shows the retrace stage with the backtracking and final optimal route through the maze between starting point B and the nearest hospital, which is hospital 6. The execution time (31.4 seconds) and the number of moves (66) to reach the destination are also shown.

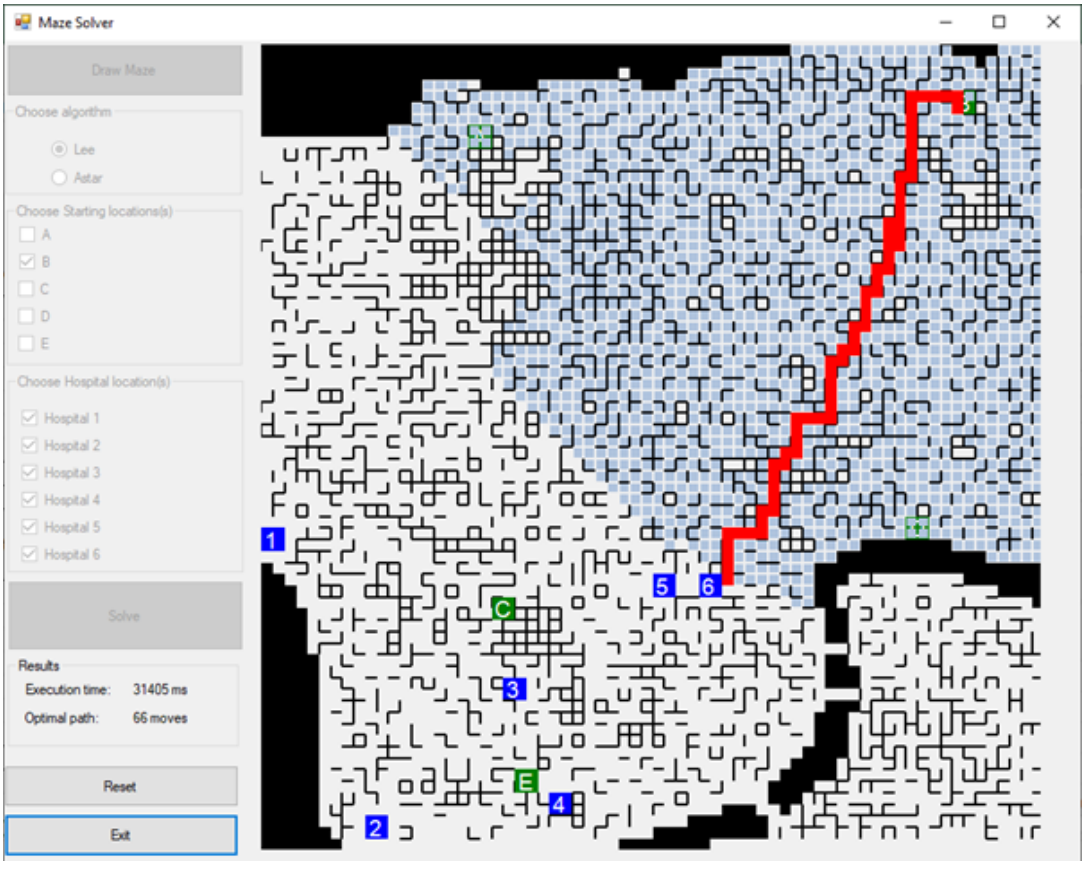

Figure 15: Optimal route generated by the Lee algorithm

6 Reflection and discussion

The use of the proposed grid-based maze technique to assist with the modelling of optimal route-finding in disaster areas proved to be reasonably easy to implement and, together with the use of the Lee algorithm, usable and positive results were obtained successfully.

Optimal paths may play a critical role in humanitarian logistics in reaching and evacuating victims to relief stations. To further highlight the capabilities of the proposed technique, all possible optimum paths between the five starting points and the six hospitals were determined using the software solution shown in Figure 15. Table 2 shows the 30 optimal routes according to the number of moves necessary to reach the destination.

Table 2: Optimal routes generated by the Lee algorithm.

Lee algorithm							
From starting point	To Hospital 1	2	3	4	5	6	To the closest hospital
A	55	87	73	80	60	64	55
B	98	114	97	99	70	66	66
C	31	29	13	26	22	29	13
D	68	72	55	57	30	22	22
E	42	17	8	4	27	31	4

Computational complexity is often an important factor, especially in situations where a time-dependent solution is critical. Under these circumstances, an acceptable trade-off level between computational time and performance may become significant, and the modeller must ensure that the time taken to solve a problem does not render the solution useless if it is not obtained timely. In this study, the modelling and solving of the problem are applied to a relatively small disaster area and computational complexity did not present a problem. To highlight the computational aspect of the Lee algorithm, the execution times (in milliseconds) were recorded for each of the possible 30 solutions and are summarised in Table 3. From the table, it can be seen that even the longest route (from starting point B to Hospital 2) is less than a minute (58 seconds).

Table 3: Execution time of the Lee algorithm.

Lee algorithm							
From starting point	To Hospital						To the closest hospital
	1	2	3	4	5	6	
A	27316	55055	41799	49091	31006	34477	27316
B	50093	58080	50132	51543	31519	31405	31405
C	15624	13753	1681	10623	6479	13661	1681
D	47919	52072	35597	37726	11319	6327	6327
E	19667	5463	1096	356	10934	13449	356

The work presented in this paper may be seen as a prototype and exploration of grid-based methods in humanitarian logistics. Consequently, some aspects may be changed to deliver even more efficient results. For example, the 70×70 grid size that was fitted over the area was determined manually – the use of geographic information systems functions, which can discretise a map, would enhance the grid placing over a disaster area. Similarly, the matrix development was performed manually, while image processing software may render a more accurate matrix representation. A more direct problem that may occur is when a starting point (victims) is situated in an area where there is no path out of the immediate vicinity. In this scenario, the algorithm would not be able to solve the optimal path problem as the maze would be seen as an invalid maze. To overcome this problem, the starting point should be moved to a position as close as possible to the original (invalid) position and then re-solved to obtain a new path that relief workers can use.

Despite the potential problems and the possible enhancements, the results obtained were reliable, accurate, and flexible regarding changing start and endpoints to determine different optimal routes. General characteristics of the Lee algorithm and its performance, which may help decision-makers when deciding on a grid-based maze solving algorithm, can be summarised as follows.

1. The algorithm is generally slower in execution time. However, if the results are not critically time-dependent, the execution time is perfectly acceptable;
2. The Lee algorithm is memory intensive as the filling stage requires the algorithm to keep track of the cells explored in the maze;
3. The algorithm is a low complexity algorithm and easy to implement; and
4. The Lee algorithm guarantees a solution if one exists.

7 Conclusion

The purpose of this paper was to offer a solution that will assist with one of the primary problems in humanitarian logistics, *i.e.*, finding the best suitable route in a disaster-stricken area. To achieve this, a grid-based maze approach, that can be solved with the Lee algorithm, was proposed. A software solution was developed, and a real-world case study was used to illustrate the feasibility of the approach. Results obtained suggest that the proposed methodology delivers valuable and useful results typically required in a humanitarian logistics scenario.

A number of contributions was made in the paper. Examples of new contributions include a grid-generating matrix representing the disaster area, that was developed. This new approach was necessitated as existing grid-generating algorithms were inappropriate because of their random generating nature which is not feasible in a disaster-damaged area. Furthermore, because the matrix method left open spaces in the grid area, the matrix algorithm was combined with an adapted version of the Kruskal algorithm to ensure that a perfect grid was generated. A user-friendly software solution was created that allows users to specify locations where victims are located and where relief or medical facilities are. The system then implements the Lee algorithm to find an optimal traversable route that may be used for evacuation purposes or to send medical or food supplies to the victims. Opportunities for further work, such as the use of GIS functionalities and image processing software, were highlighted in the reflection and discussion section.

Appendix

The complete 70×70 matrix representing the damage to the New Orleans area is presented in this Appendix. The matrix is used to generate a maze for the New Orleans disaster area. See Algorithm 2 in Section 5.2.

{4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,3,4,4,4,4,0,0,0,0,0,0},
{4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,3,0},
{4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,4,4,4,4,4,0,0,0,0,0,0,0,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,1},
{4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,1,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,0,0,0,0,0,0,0},
{4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,1,0,0,0,0,0,1,2,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,3,3,1,0,0,0,0,0,0},
{4,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,2,0,0,3,1,0,1,0,0,0,0,3,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,3,0,0,1,0,2,3,1,0,3,0,0,0,2,3,1,0,0,0,0,0,0},
{4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,1,0,0,0,1,2,2,0,0,2,0,0,3,2,0,0,0,0,0,3,0,0,1,0,0,0,1,0,0,3,1,2,0,0,2,0,0,1,1,3,0,1,0,0,0,0,0,3,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,1,0,0,3,0,2,1,0,2,3,1,0,0,1,3,1,0,1,0,0,3,2,0,0,0,0,0,0,1,0,0,2,0,0,3,0,0,0,3,0,0,0,2,0,2,0,3,3,0,0,0,0,3,0,0,0,0,0,0},
{0,0,0,0,0,0,0,3,3,1,2,0,0,0,1,2,2,0,3,3,3,1,3,2,3,3,0,2,0,0,0,2,1,0,0,0,0,0,0,1,0,3,3,3,1,3,1,2,0,0,0,0,0,3,0,0,1,2,3,1,0,0,0,2,1,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,1,2,3,2,3,3,2,0,3,0,0,0,0,0,0,0,0,0,0,0,0,2,0,3,2,2,2,1,3,3,1,0,2,0,0,0,2,0,2,1,3,3,3,0,0,3,1,1,0,0,0,0,0},
{1,0,0,1,0,0,0,1,0,3,1,1,0,0,3,1,2,1,3,3,3,2,1,0,2,2,1,0,0,0,0,0,0,0,0,0,3,1,0,2,2,0,3,1,0,3,2,1,0,0,1,3,1,0,0,3,0,1,2,3,1,0,2,3,1,0,0,0,0,0},
{2,0,0,0,0,2,0,0,2,2,3,3,1,0,0,0,1,0,1,3,3,2,0,3,3,3,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,2,0,0,3,3,0,2,2,2,0,1,0,3,2,0,2,3,2,2,3,3,0,0,0,0,0},
{0,0,3,0,0,0,1,1,0,3,1,2,0,0,3,3,1,2,3,3,1,1,2,2,2,3,1,0,0,0,0,0,0,0,0,0,1,2,0,3,3,1,0,2,1,0,2,0,0,0,0,0,0,2,0,2,0,0,2,2,1,2,3,1,1,0,0,0,0,0},
{0,0,0,0,0,0,2,2,0,3,0,1,1,0,2,0,3,0,2,3,1,2,3,0,3,1,0,0,0,0,0,0,0,0,0,0,2,0,0,3,3,2,1,0,0,2,1,2,0,0,0,0,0,3,1,2,0,0,2,3,3,3,3,3,3,0,0,0,0,0},
{0,0,0,0,0,0,0,3,3,0,2,1,3,3,2,1,0,1,2,2,2,0,3,3,1,0,0,0,0,0,0,0,0,0,0,0,2,2,0,1,3,1,0,0,0,1,0,3,1,0,1,1,3,2,2,0,0,0,0,3,2,3,3,3,3,2,3,1,0,0},
{0,0,0,0,0,0,0,2,0,0,2,1,1,2,2,0,2,0,1,0,3,1,1,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,1,0,0,0,1,0,2,0,0,3,1,0,2,1,0,2,3,1,0,2,3,3,2,2,0,0,0,0,0,0},
{3,0,0,0,0,0,0,0,0,2,0,2,0,3,3,3,3,1,0,3,3,3,1,0,2,2,0,0,0,0,0,0,0,0,0,0,1,0,0,2,0,0,2,0,0,2,0,1,0,2,0,0,0,2,2,0,0,0,0,0,2,0,0,0,0,2,0,0,0,0},
{0,0,0,0,0,1,0,0,0,1,2,0,0,2,2,3,3,0,1,2,3,3,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,1,0,0,0,0,0,0,0,0,0,0,0,0,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,3,3,3,3,2,3,3,3,1,0,0,0,0,0,0,0,0,0,1,2,1,0,3,0,0,0,0,0,0,0,0,0,3,1,0,2,0,1,0,0,0,2,1,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,2,3,3,3,2,3,3,3,0,3,2,3,3,2,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,1,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,3,3,3,1,3,2,1,3,0,3,3,3,1,0,0,0,0,0,0,0,0,0,0,0,0,3,1,0,0,1,0,0,0,0,0,0,0,0,2,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,2,0,2,0,0,0,0,0,2,2,2,2,0,0,3,3,3,2,3,1,3,2,0,0,0,0,0,0,0,0,0,0,0,0,3,1,2,1,0,3,3,1,0,0,0,0,0,2,0,0,0,0,3,3,1,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,3,1,0,0,0,2,1,1,1,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,2,0,0,2,0,0,0,0,0,0,0,3,1,0,3,3,0,0,0,2,1,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,1,0,0,0,2,1,2,1,0,3,1,2,2,3,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,1,0,3,0,3,2,0,2,2,2,0,3,0,0,0,0,0,0,2,0,0,0,0,0},
{0,0,0,2,0,2,0,0,0,0,0,0,3,3,1,0,2,2,0,3,3,3,3,3,1,0,0,0,3,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,0,0,3,0,2,1,0,1,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,3,0,0,0,0,0,2,2,0,0,2,2,0,2,3,3,2,2,2,0,0,0,0,0,0,0,0,0,2,0,0,2,2,0,0,1,0,0,0,1,0,0,2,1,0,0,0,0,0,0,0,2,0,0,0,1,0,0,1,0,1,0,0,0},
{0,0,0,0,0,0,2,0,1,0,0,0,2,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,1,0,3,0,0,0,2,2,0,3,1,0,2,0,0,0,0,0,0,0,0,0,0,2,0,0,1,1,2,0,0,0,0},
{0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,0,0,2,0,0,2,0,0,3,3,3,1,0,0,0,0,0,1,0,0,0,1,0,0,2,0,2,0,0,0,0,0,0},
{0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,1,0,0,0,3,1,1,0,0,0,0,2,0,0,2,0,0,1,0,0,0,0,1,0,0,0,0,0},
{0,1,2,2,0,3,3,1,0,0,0,0,0,0,0,0,2,0,0,0,0,2,0,2,0,1,0,0,3,2,0,0,0,0,0,0,0,0,0,0,3,0,0,2,2,1,0,3,0,1,0,0,3,1,3,1,0,0,0,0,2,0,0,0,2,3,0,0,0,0},
{3,0,0,0,0,2,2,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,1,0,0,1,0,0,3,1,3,0,0,0,2,1,3,1,0,1,0,0,0,0,0,0,2,2,3,0,0,0,3,0,1,0,3,2,1,0,0,0,1,0,0,0,1,0,0,0},
{0,1,2,0,0,0,0,3,0,0,2,0,0,0,0,0,0,0,0,0,0,3,1,3,1,0,0,0,2,0,1,1,0,2,0,0,3,1,2,1,3,1,1,2,3,2,0,0,2,0,3,2,0,2,1,1,2,0,0,0,0,1,0,0,0,2,0,0,0,0},
{3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,3,0,0,3,1,0,0,0,0,3,0,1,0,0,2,3,2,0,1,0,2,0,1,0,2,0,0,0,3,0,0,2,2,1,0,0,0,0,2,2,0,1,0,0,3,1,0,0,0,0},
{2,2,0,0,0,0,0,2,0,0,0,3,0,0,0,0,0,0,0,0,2,0,1,2,0,2,0,0,0,0,0,0,1,3,3,2,0,2,1,0,0,0,3,3,0,0,1,2,2,2,3,3,3,3,2,2,2,0,0,0,3,3,2,0,0,2,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,1,0,2,2,3,1,0,0,3,0,0,0,0,2,0,0,1,0,2,0,2,0,0,0,0,2,3,2,0,0,3,3,2,2,3,0,0,1,0,0,0,1,3,2,1,0,0,0,1,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,2,0,0,1,0,2,0,3,1,0,1,2,0,0,0,1,0,1,0,0,0,0,2,3,0,3,0,1,2,3,2,3,3,2,0,1,0,1,0,2,1,0,0,0,1,2,2,0,0,2,2,0,0,0,3,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,3,1,1,3,3,1,2,0,0,0,0,0,0,3,1,2,0,0,0,0,0,0,2,0,1,2,0,3,3,2,3,0,1,3,3,3,1,3,1,0,2,0,2,0,0,0,0,0,0,0,0,2,3,2,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,2,3,2,2,1,1,0,0,0,0,0,3,2,2,0,0,0,2,0,2,1,0,0,0,0,0,3,1,2,0,2,3,3,2,2,2,0,0,0,2,1,0,0,2,0,0,0,0,0,0,2,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,2,0,0,0,3,3,0,3,1,1,0,0,0,0,0,0,3,1,3,3,0,0,0,2,0,3,1,0,1,0,0,0,2,1,0,3,2,2,0,3,1,0,1,3,0,3,1,0,0,0,1,0,0,0,2,0,3,3,1,0},
{0,0,0,3,1,0,0,0,2,3,3,1,2,1,2,0,0,2,0,2,0,0,0,0,0,0,2,0,2,2,0,0,1,0,0,2,0,2,0,2,2,0,3,3,0,3,0,0,3,0,0,2,2,3,1,1,0,3,1,0,0,0,3,1,0,0,2,2,0,0},
{0,0,0,2,0,0,0,0,0,2,2,2,3,0,1,2,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0,2,0,0,1,0,2,0,0,0,0,2,3,1,0,3,0,0,2,0,3,1,3,0,1,0,0,2,0,0,0,2,2,2,0,0,2,0,0,1},
{0,0,0,0,0,0,0,0,0,0,0,2,0,2,3,2,1,0,0,3,1,0,0,0,3,1,3,0,0,1,0,0,0,0,3,3,0,1,3,0,2,0,0,0,0,2,3,1,0,0,0,1,0,0,2,0,0,0,0,0,0,1,0,2,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,2,1,2,2,3,3,3,2,2,1,3,1,2,0,2,0,2,0,0,0,0,0,2,2,0,0,2,0,0,1,0,0,0,0,3,3,3,1,2,0,0,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,2,1},
{0,0,0,0,0,0,0,3,0,0,0,0,0,0,3,2,2,2,3,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,2,2,2,0,0,4,4,4,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,2,0},
{4,0,0,0,0,0,1,0,0,0,3,3,1,0,0,2,0,3,0,2,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,3,1,0,0,2,2,0,0,0,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,0,0},
{4,4,0,0,0,1,2,2,2,0,1,2,1,0,3,0,0,2,2,0,2,2,1,0,0,1,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,4,4,0,0,1,0,0,0,0,0,0,4,4,4,4,4,4,4,4,4},
{4,4,4,0,0,3,3,1,0,3,3,1,1,0,2,1,3,1,0,3,2,2,0,0,1,2,0,0,0,3,1,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,4,4,0,0,2,2,0,0,0,2,0,0,2,0,0,0,4,4,4,4,4,4},
{4,4,4,0,0,2,2,0,0,2,2,2,3,3,0,1,2,0,0,1,3,0,1,2,1,0,0,3,1,2,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,4,0,0,0,0,0,0,3,1,0,0,0,0,0,0,0,0,4,4,4},
{4,4,4,4,0,0,0,0,0,0,0,3,1,2,0,1,3,1,0,2,2,3,3,3,3,1,2,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,4,4,0,0,0,0,1,0,2,0,0,0,0,0,0,0,0,0,0,0,0,4},
{4,4,4,4,0,0,0,0,0,0,0,3,2,0,3,1,2,2,3,0,3,2,3,1,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,4,0,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,0,0,0,1,0,2,0,3,1,0,0,3,1,0,2,0,2,3,3,3,3,3,0,0,0,2,0,0,2,1,0,3,1,0,0,0,0,0,0,0,0,0,0,0,4,4,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,0,0,0,0,1,0,0,3,1,1,0,2,0,0,2,0,3,2,3,3,2,3,3,0,1,0,2,1,2,0,1,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,0,0,1,2,0,2,2,0,3,2,0,2,1,0,2,2,3,2,2,3,3,2,2,1,0,0,0,0,2,0,3,1,2,0,0,0,0,0,0,0,0,2,1,4,4,0,0,2,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,0,1,2,0,0,0,0,0,0,0,2,1,0,0,0,2,1,2,0,1,2,3,0,0,2,2,2,3,3,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,4,4,0,0,0,2,0,0,0,0,0,1,0,0,0,0,0,0,0},
{4,4,4,4,4,0,2,1,0,2,0,2,0,0,0,0,0,2,1,3,0,0,0,1,0,0,3,1,0,2,0,2,2,2,1,0,2,0,0,0,0,0,0,0,0,0,0,0,0,4,4,4,0,0,0,0,0,1,0,0,0,0,0,3,1,0,0,0,0,0},
{4,4,4,4,4,0,0,2,1,0,0,0,0,0,0,1,0,0,1,2,0,0,0,0,3,3,1,1,0,0,1,1,2,1,2,0,0,0,0,0,1,3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,2,1},
{4,4,4,4,4,0,0,2,3,0,0,0,1,0,0,2,0,2,1,2,0,2,1,0,2,0,2,3,1,3,1,2,0,3,3,0,1,0,0,2,2,0,1,0,0,0,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0},
{4,4,4,4,4,0,2,0,1,0,0,0,0,0,0,0,0,0,3,1,0,0,0,3,0,2,3,3,1,2,2,0,0,2,1,2,2,0,0,0,0,2,0,0,0,1,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,0,0,2,0,2,1,0,0,1,0,0,0,1,0,0,1,1,2,0,0,2,2,0,2,0,3,3,3,3,0,2,0,0,0,1,1,0,3,1,2,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
{4,4,4,4,4,0,0,0,3,1,0,0,0,1,0,2,1,0,2,0,2,0,0,0,1,0,1,3,1,0,0,3,3,1,3,1,0,0,0,2,2,0,1,2,0,0,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,3,0},
{4,4,4,4,4,0,0,2,2,0,0,0,0,2,0,0,2,0,0,2,0,0,1,0,3,2,0,2,0,2,2,0,2,0,2,0,0,0,0,1,1,0,0,0,0,0,0,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0},
{4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,1,0,0,0,2,0,2,0,3,1,2,0,0,0,0,2,0,0,0,0,1,0,0,2,0,0,0,0,0,0,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,0,2,0,0,0,0,0,0,0,2,1,1,0,0,0,3,2,0,0,2,0,0,0,0,0,0,0,0,0,0,0,1,2,1,0,0,0,0,0,0,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0,0},
{4,4,4,4,4,0,0,0,0,0,0,3,1,3,1,2,3,0,1,2,0,0,0,0,0,0,0,0,1,2,0,0,0,0,2,2,0,2,0,1,0,0,0,4,4,4,4,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,0,0,0,0,0,0,2,0,2,0,3,3,3,3,2,0,0,1,1,0,0,3,1,2,0,0,0,2,0,0,3,1,0,2,0,0,0,0,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0},
{4,4,4,4,4,0,0,0,0,0,0,2,2,0,2,2,2,0,2,0,1,2,1,2,0,0,3,1,0,0,0,1,0,0,0,2,0,0,0,4,4,4,0,0,4,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,4,4,4,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0,0,1,3,2,3,1,0,0,0,1,0,0,0,0,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
{4,4,4,4,4,4,0,0,0,0,0,2,1,0,0,0,0,1,0,0,0,0,2,0,0,0,2,0,0,0,0,0,0,4,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0},
{4,4,4,4,4,4,4,0,0,0,0,0,2,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,4,4,4,4,4,4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0}.

References

[1] Adamatzky, A. 2012, *Slime mold solves maze in one pass, assisted by gradient of chemo-attractants.* IEEE Transactions on Nanobioscience, 11(2):131-134.

[2] Ahmed, W., Najmi, A., Khan, F. & Aziz, H. 2019, *Developing and analyzing framework to manage resources in humanitarian logistics.* Journal of Humanitarian Logistics and Supply Chain Management, 9(2):270-291.

[3] Ashlock, D., Lee, C. & McGuinness, C. 2011, *Search-based procedural generation of maze-like levels.* IEEE Transactions on Computational Intelligence and AI in Games, 3(3):260-273.

[4] Aki, O. & G ull u, A. 2016, *Obtaining path data from maze image using image processing techniques.* In Conference proceedings. International Scientific Conference (UNITECH2016), Gabrovo, Bulgaria. Gabrovo, UNITECH, 326-329.

[5] Anuar, W.K., Moll, M., Lee, L.S., Pickl, S. & Seow, H.V. 2019, *Vehicle routing optimization for humanitarian logistics in disaster recovery: A survey.* International Conference, Security and Management (SAM19).

[6] Barnouti, N.H., Al-Dabbagh, S.S.M. & Naser, M.A.S. 2016, *Pathfinding in strategy games and maze solving using a search algorithm.* Journal of Computer and Communications, 4(11):15-25.

[7] Batista e Silva, F., Gallego, J. & Lavalle, C. 2013, *A high-resolution population grid map for Europe.* Journal of Maps, 9(1):16-28.

[8] Bekker, J.F. & Schmid, J.P. 2006, *Planning the safe transit of a ship through a mapped minefield*, ORiON, 22(1):1-18.

[9] Bìl, M., Vodàk, R., Kubeček, J., Bìlovà, M. & Sedonìk, J. 2015, *Evaluating road network damage caused by natural disasters in the Czech Republic between 1997 and 2010.* Transportation Research Part A: Policy and Practice, 80:90-103.

[10] Boustan, L.P., Kahn, M.E., Rhode, P.W. & Yanguas, M.L. 2020, *The effect of natural disasters on economic activity in US counties: A century of data.* Journal of Urban Economics, 118:1-26.

[11] Bollobás, B. 2013, *Modern graph theory.* New York, NY: Springer Science & Business Media.

[12] Calma, J. 2020, What you need to know about the Australia bushfires. https://www.theverge.com/2020/1/3/21048891/australia-wildfires-koalas-climate -change-bushfires-deaths-animals-damage Date of access: 28 Feb. 2020.

[13] Chaudhari, A.M., Apsangi, M.R. & Kudale, A.B. 2017, *Improved A-star algorithm with least turn for robotic rescue operations.* In Mandal J., Dutta P. & Mukhopadhyay S. (eds) Computational Intelligence, Communications, and Business Analytics. CICBA 2017, Communications in Computer and Information Science, vol 776. Springer, Singapore.

[14] Daneshjo, N., Kralik, M., Petrovcikovà, K., Pajerskà, E.D. & Paštéka, M. 2019, *Avoiding the obstacles in the robot working zone by using the Lee algorithm.* Coordinates, 13(2):72-83.

[15] DeWaard, J., Curtis, K.J. & Fussell, E. 2016, *Population recovery in New Orleans after Hurricane Katrina: exploring the potential role of stage migration in migration systems.* Population and Environment, 37(4):449-463.

[16] Dijkstra, E.W. 1959, *A note on two problems in connexion with graphs.* Numerische Mathematik, 1(1), 269–271.

[17] Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T. & Jurišica, L. 2014, *Path planning with modified A star algorithm for a mobile robot.* Procedia Engineering, 96:59-69.

[18] Duran, S., Ergun, Ö., Keskinocak, P. & Swann, J.L. 2013, *Humanitarian logistics: advanced purchasing and pre-positioning of relief items.* In James, H. eds. *Handbook of Global Logistics.* New York: Springer, 447-462).

[19] Fankhauser, P. & Hutter, M. 2016, *A universal grid map library: Implementation and use case for rough terrain navigation.* In Koubaa, A. eds. *Robot operating system (ROS).* Cham, Germany: Springer, 99-120.

[20] Foltin, M. 2011, *Automated maze generation and human interaction.* Masaryk University (Thesis – PhD).

[21] Gibbens, S. 2019, *Hurricane Sandy, explained.* https://www.nationalgeographic.com/environment/natural-disasters/reference/hurricane-sandy/ Date of access: 12 Dec 2019.

[22] Gordon, V.S. & Matley, Z. 2004, *Evolving sparse direction maps for maze pathfinding.* In Conference proceedings, Congress on Evolutionary Computation (04TH8753), Portland, USA. New York: IEEE, 835-838).

[23] Griffith, D.A., Boehmke, B., Bradley, R.V., Hazen, B.T. & Johnson, A.W. 2019, *Embedded analytics: improving decision support for humanitarian logistics operations.* Annals of Operations Research, 283:247-265.

[24] Gupta, B. & Sehgal, S. 2014, *Survey on techniques used in autonomous maze solving robot.* In Conference proceedings: 5th International Confluence - The Next Generation Information Technology Summit (Confluence 2014), Noida, India. New York: IEEE, 323-328.

[25] Harabor, D.D. & Grastien, A. 2011, *Online graph pruning for pathfinding on grid maps.* In Conference proceedings: 25th AAAI Conference on Artificial Intelligence. San Francisco, USA. California: AAAI. pp. 1114-1119.

[26] Hart, P.E., Nilsson, N.J. & Raphael, B. 1968, *A formal basis for the heuristic determination of minimum cost paths.* IEEE Transactions on Systems Science and Cybernetics, 4(2):100-107.

[27] Hartomo, K., Ismanto, B., Nugraha, A., Yulianto, S. & Laksono, B. 2019, *Searching the shortest route to distribute disaster's logistical assistance using Dijkstra method.* 4th Annual Applied Science and Engineering Conference, Journal of Physics: Conference Series. 1402, 1-7.

[28] Hendrawan, Y.F. 2020, *Comparison of hand follower and dead-end filler algorithm in solving perfect mazes.* Journal of Physics: Conference Series, 1569(2):22-59.

[29] Jabbour, C.J.C., Sobreiro, V.A., Lopes de Sousa Jabbour, A.B., De Souza Campos, L.M., Mariano, E.B. & Renwick, D.W.S. 2019, *An analysis of the literature on humanitarian logistics and supply chain management: paving the way for future studies.* Annals of Operations Research, 283:289–307.

[30] Jahre, M., Persson, G., Kovács, G. & Spens, K.M. 2007, *Humanitarian logistics in disaster relief operations.* International Journal of Physical Distribution & Logistics Management. 37(2):99-114.

[31] Jannu, S. & Jana, P.K. 2016, *A grid based clustering and routing algorithm for solving hot spot problem in wireless sensor networks.* Wireless Networks, 22(6):1901-1916.

[32] Jubair, F. & Hawa, M. 2020, *Exploiting obstacle geometry to reduce search time in grid-based pathfinding.* Symmetry, 12(7):1186.

[33] Jordans, F. 2022, *The Times of Israel, Hurricane Ida, Europe Floods made 2021 a costly year of disasters.* https://www.timesofisrael.com/hurricane-ida-europe-floods-made-2021-a-costly-year-of-disasters/ Date of access: 26 May 2020.

[34] Kim, J.H., Min, K.S. & Yeo, W.Y. 2014, *A design of irregular grid map for large-scale Wi-Fi LAN fingerprint positioning systems.* The Scientific World Journal, 1-13.

[35] Kim, P.H. 2019, *Intelligent Maze Generation. Ohio: The Ohio State University.* (Thesis – PhD).

[36] Kruskal, J.B. 1956, *On the shortest spanning subtree of a graph and the traveling salesman problem.* Proceedings of the American Mathematical society, 7(1):48-50.

[37] Law, G. 2013, *Quantitative comparison of flood fill and modified flood fill algorithms.* International Journal of Computer Theory and Engineering, 5(3):503-508.

[38] Lee, C.Y. 1961, *An algorithm for path connections and its applications.* IRE Transactions on Electronic Computers, 3:346-365.

[39] Lee, H.L., Lee, C.F. & Chen, L.H. 2010, *A perfect maze based steganographic method.* Journal of Systems and Software, 83(12):2528-2535.

[40] Lerch, J.P., Yiu, A.P., Martinez-Canabal, A., Pekar, T., Bohbot, V.D., Frankland, P.W., Henkelman, R.M., Josselyn, S.A. & Sled, J.G. 2011, *Maze training in mice induces MRI-detectable brain shape changes specific to the type of learning.* Neuroimage, 54(3): 2086-2095.

[41] Lukosch, H. & Comes, T. 2019, *Gaming as a research method in humanitarian logistics.* Journal of Humanitarian Logistics and Supply Chain Management, 9(3):352-370.

[42] Massazza, A., Brewin, C.R. & Joffe, H. 2019, *The nature of "natural disasters": survivors' explanations of earthquake damage.* International Journal of Disaster Risk Science, 10(3):293-305.

[43] Mirahadi, F. & McCabe, B.Y. 2020, *EvacuSafe: A real-time model for building evacuation based on Dijkstra's algorithm.* Journal of Building Engineering, 34(2021):1-14.

[44] Munyaka, J.C.B. & Yadavalli, V.S.S. 2020, *Decision support framework for facility location and demand planning for humanitarian logistics.* International Journal of System Assurance Engineering and Management, 11(5):1-20.

[45] Nelson, M.J. & Smith, A.M. 2016, *ASP with applications to mazes and levels.* In Shaker, N., Togelius, J., & Nelson, M.J. eds. *Procedural content generation in games.* Switzerland, Cham: Springer. 143-157.

[46] Olano, G. 2019, *After the storm: Japan's recovery efforts post-Hagibis.* https://www.insurancebusinessmag.com/asia/news/breaking-news/after-the-storm-japans-recovery-efforts-posthagibis-189234.aspx Date of access: 20 Dec. 2019.

[47] Pershin, Y.V. & Di Ventra, M. 2011, *Solving mazes with memristors: a massively parallel approach.* Physical Review E, 84(4):1-7.

[48] Polanczyk, M., Strzelecki, M. & Slot, K. 2012, *Lee-algorithm based path replanner for dynamic environments.* In Conference proceedings: 2012 International Conference on Signals and Electronic Systems (ICSES 2012). Wroclaw, Poland. New York: IEEE. pp. 1-4).

[49] Prim, R.C. 1957, *Shortest connection networks and some generalizations.* Bell System Technology Journal, 36(6):1389–1401.

[50] Qureshi, A.G. & Taniguchi, E. 2020. *A multi-period humanitarian logistics model considering limited resources and network availability.* Transportation Research Procedia, 46(2020):212–219.

[51] Reddy, A.V., Vinoth, G. & Chiranjeevi, G.N. 2018, *Implementation of Lee's algorithm for different routing constraints.* In Conference proceedings: 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT 2018), Bengaluru, India. New York: IEEE, 2488-2491.

[52] Ritchie, H. & Roser, M. 2019, *Natural Disasters.* https://ourworldindata.org /natural-disasters Date of access: 5 Nov 2020.

[53] Rodríguez-Espíndola, O., Albores, P. & Brewster, C. 2018, *Disaster preparedness in humanitarian logistics: a collaborative approach for resource management in floods.* European Journal of Operational Research, 264 (3):978-993.

[54] Shaluf, I.M. 2007, *An overview on disasters.* Disaster Prevention and Management: An International Journal, 16(5):687-703.

[55] Shao, J., Wang, X., Liang, C. & Holguín-Veras, J. 2020, *Research progress on deprivation costs in humanitarian logistics.* International Journal of Disaster Risk Reduction, 42:1-12.

[56] Shavarani, S.M. 2019, *Multi-level facility location-allocation problem for post-disaster humanitarian relief distribution.* Journal of Humanitarian Logistics and Supply Chain Management, 9(1):70-81.

[57] Sigala, I.F. & Wakolbinger, T. 2019, *Outsourcing of humanitarian logistics to commercial logistics service providers.* Journal of Humanitarian Logistics and Supply Chain Management, 9(1):47-69.

[58] Sturtevant, N.R. 2012, *Benchmarks for grid-based pathfinding.* IEEE Transactions on Computational Intelligence and AI in Games, 4(2):144-148.

[59] Sydnor, S., Niehm, L., Lee, Y., Marshall, M. & Schrank, H. 2017, *Analysis of post-disaster damage and disruptive impacts on the operating status of small businesses after Hurricane Katrina.* Natural Hazards, 85(3), 1637-1663.

[60] United Nations. 2022. https://unric.org/en/2021-floods-un-researchers-aim-to-better-prepare-for-climate-risks/ Date of access: 12 Mar 2020.

[61] Walker, R.J. 1960, *An enumerative technique for a class of combinatorial problems.* In American Math Society, eds. Conference proceedings. Proceedings of Symposia in Applied Mathematics, Indonesia. Providence, American Mathematical Society, 91-94.

[62] WHO. 2020. World Health Organisation https://www.who.int/hac/about/definitions/en/ Date of access: 13 April 2020.

[63] Widera, A., Lechtenberg, S., Gurczik, G., Bähr, S. & Hellingrath, B. 2017, *Integrated logistics and transport planning in disaster relief operations.* In Comes, T., Bénaben, F., Hanachi, C., Lauras, M., & Montarnal, A., eds. Conference proceedings. 14th International Conference on Information Systems for Crisis Response and Management. Albi, France. 752-764.

[64] Xu, J., Wang, Z., Shen, F., Ouyang, C. & Tu. Y. 2016, *Natural disasters and social conflict: A systematic literature review.* International Journal of Disaster Risk Reduction. 17:38-48.