Emergency Escape Routing and Evacuation using Bellman Ford's Algorithm

Baby D. Dayana, Shiv Pratap Singh, Shraman Das, Pankaj Gautam

Abstract: Evacuation Technique which addresses routing the populaces within the endangered architecture to the bordering emergency-exit points minimizing the mortality rate and avoiding hazards and obstacles at the same time.The fundamental magnitude would be to minimize the risk of total mortality during an emergency condition.The superlative case scenario would be to relinquish all individuals with smidgeon panic and naught transience tempo.This scrutiny will lessen the evacuation time factor by using the deluxe appropriate algorithm available for the shortest path prediction with all of the constraints.

Keywords: Bellman Ford's Algorithm,Emergency evacuation,Iot for evacuation,Raspberry Pi,Shortest path planning.

I. INTRODUCTION

During emergency situations, its of prime importance to calculate the shortest distance of the person in emergency to the fire exit whilst keeping in mind the hazards of fire and blockage. However, there are many algorithms which focus on the shortest path from the source node to the destination node minimizing the mortality rate.But none of the algorithms take into account, the negative weights which can be present during an evacuation. The significance of a negative weight is that while traversing other edges we might have to go pay a cost, but while traversing a negative edge, we might get some benefit rather than paying a cost for the travelling.

Some of the examples are:

1. If the elevator in that direction is still working then, instead of moving to a fire exit on the 15th floor, the person can directly move to the ground floor of the exit.
2. If a corridor is on fire but the entrance of the corridor has a fire extinguisher then the person can label that corridor as a negative weight because once he extinguishes the fire with the extinguisher the corridor which previously was untraversable would be easily traversable now.
3. The path to the exit is a bit longer than usual but the corridor is wider and it would prevent panic and bottlenecking.

The project would be to simplify the escape plan algorithm and pose an efficient evacuation system during emergency. There would be environment sensors(particularly smoke, temperature and water level sensors) installed in the respective rooms during the development of the building based on the emergency situation that the algorithm is trying to design evacuation paths.The EVACNET+ software would be used and the model of the entire building would be obtained and fed to the module in the form of graphs and edges.

During the emergency, the module would automatically use the algorithm(i.e Bellman Fords) and route the shortest distance to the nearest fire exit to the user's mobile device.
The information would be transmitted to the mobile devices by using a common wifi hotspot and a router.
All the devices connected to the particular wifi would be able to route the user to the nearest fire exit.
However special evacuation users as deemed previously during fire drills would be going through all the corridors and if he finds a path that would be shorter and is not detected by the model previously, he would send a request to the model to assign a negative weight to the path so that the model recalculates the distance and more people would be able to use that path and reach the nearest exit. This method would split the mass and also prevent bottlenecking situations during evacuation.

II. RELATED WORK

Till date there are a lot of papers submitted which address the minimum time and shortest path to the exits in a building.

Using artificially intelligent methods coupled with Iot has already been implemented in a lot of buildings as mentioned by Huxian Jiang[1]. He has used Ant Colony Optimization algorithm for his evacuation model.When no fire outbreak is observed i.e, during static conditions the best path followed by the ants is by avoiding the static obstacles. But during a dynamic evacuation algorithm, the algorithm is optimized to track and observe the relative dangers of smoke and fire whilst determining the shortest path to the exit[1]

A brief comparison between two algorithms of Bellman Ford and Dijkstra's has been made where the observations clearly state that Dikstra's Algorithm performs better than Bellman Ford algorithm when the graph has non-negative weights assigned to it. But Dijkstra's Algorithm fails to notice if there are any negative weight cycles in the graph if negative weights are present.[2]

It clearly mentions that if the graph has negative weights then Bellman Ford's Algorithm would be the best Algorithm to find the shortest route between two points.

However, if positive weights are used then Dijkstra's would perform better than Bellman Ford's.

Revised Manuscript Received on June 30, 2020.
* Correspondence Author

Baby D Dayana*, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India.
Shraman Das, Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India.
Shiv Pratap Singh, Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India.
Pankaj Gautam, Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Tamil Nadu, India.

If lower load with higher response is desirable then Dijkstra's would be a better choice[2]

Usage of EVACNET to get the graphical model of the building and then using Single SN Single ED algorithms to update the weights of the graphs has been implemented byXingxuan Wang and Huan Liu[4] where the distance of the evacuees is taken into account during evacuation.The dynamic capacity of the corridors and paths are also calculated.

III. PROPOSED WORK:

The proposed solution to a safe and efficient way of evacuating the buildings is:

First the entire floor plan is updated to the module which is going to calculate the shortest path to the module by using a software like EVACNET.

EVACNET uploads the entire plan of the building to our computer.

The plan has the set of nodes which represent the rooms,offices and the staircases and these are connected by arcs which have the value of their weight mentioned on them.

After this value is completed, and all the nodes are updated with the model, the model is going to get feedback from mainly two portions namely

Users (information about bottlenecks and other inconveniences)

Sensors (water,smoke,and fire sensors should relay the information)

During an emergency condition, the model would relay the shortest path to each users mobile device through which they would be able to understand the map and follow the shortest path outside.

However, if there is a better path available and a person discovers it whilst evacuating,the person should have the ability to update the data to the module which would in turn update the same data to the model and label the edge as a negative weight.

This would greatly improve the accuracy and efficiency of the model as it would open up multiple ways to the many fire exits located in the building.

However, if there are negative cycles in the group due to the updating of the edge values the system will reset and show the original path just before the last update of the edge.

In order to avoid confusion and frequent system resets, frequent emergency drills should be

conducted in the buildings and the access to change the weights of the models should only be given to people who perform calmly during the drills.

This would prevent the model from getting incorrect inputs from the people as much as possible.

Now. the model would have to be a small computer, anything that can transmit receive and compute without taking too much cost.
One of the most important devices to note are Raspberry Pi's.

The entire algorithm is to be fed to the Pi and connections are established according to the need of the building based on the size and structure mainly.

To do this we would write a python program implementing the algorithm with inputs as the weights from the EVACNET module and the entire graph should be fed to the computer.
The Pi's are strategically placed throughout the building to ensure efficient transmission of data to each and every user.

The Pi should also be configured to send an emergency SOS message to the nearest Emergency Relief Forces as soon as possible.

The modules under use are

An EVACNET module

The Sensor's metric data

The computer(Raspberry Pi)

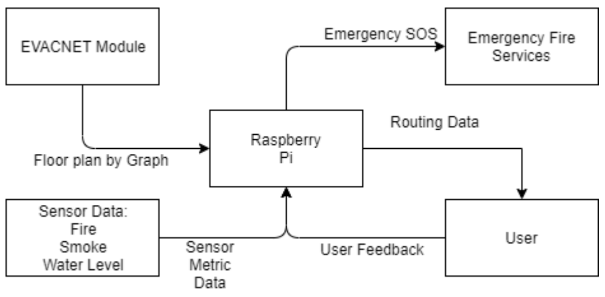


Fig 1:Architecture Diagram

IV. EVACNET:

Generally EVACNET is a standalone system for fire evacuation in buildings. But, the model would only use the floor plan returned as a graph which would serve as an input to the algorithm[4]. The weights of the edges would be initialized as infinity at first. But as soon as the inputs from the sensors would be fed into the program the graph would be made.

While discussing the points of the EVACNET module it's important to discuss how the weights of the graph would be assigned.

The weights of the graph would be assigned according to the likeliness of an individual to traverse the path during an emergency. For example, during a fire outbreak, users are far less likely to traverse a corridor blazing with fires. Hence the particular path would be assigned a high weight/cost. However, paths which can benefit the users are used as negative weights. For example, a user can deem a path to be a negative weight bearing edge if there is a small fire which can be easily extinguished and used as an alternate pathway to an exit. This is actually a debatable topic as to when to declare an edge to be negative in cost. EVACNET also calculates the number of people in each node so that emergency services are easy to avail.

However, the input to the algorithm about the graph can also be done manually.

V. SENSORS:

The prime module of this project would be the sensors continuously relaying the information to the Pi as metric data throughout the evacuation process.

Some of the sensors that are gonna be a part of the evacuation can be:

Smoke Sensor(MQ-2)- It can detect smoke,fire and harmful gases such as methane,LPG,etc.

Heat Sensors(LM35)- The output voltage of the sensor is directly proportional to the amount of heat sensed.

Water Level Sensor Float Switch(P43)- The sensor detects the water level in a room. The sensor can be tweaked to trigger an emergency response as soon as the water level rises to a dangerous level.

VI. THE PI:

This is the heart of the model as all the receiving and transmitting would be done by this Pi. It has 3 main modules

Receiver- Receives metric data from sensors and feeds to the input of the algorithm. Also receives input from user about the negative weights and transfers it to the algorithm input.

Transmitter- Transmits the data through a wifi/bluetooth module(Any GSM/GRS/GNSS bluetooth transmitter) would work. The main function of the transmitter is to relay the information as fast as possible to the users using the software.

Algorithm- Bellman Ford's algorithm takes input from the EVACNET module, a graph with no edge weights. After that the algorithm is fed the details by the metric values of the sensors. Then the algorithm returns the shortest path from each node to the exit node for each user.

Connecting two or more Raspberry computers would be required if the building is large and bluetooth range of one individual Pi doesn't cover the entire building. This can be made possible by XBee or any other wireless module that can be plugged into a Raspberry Pi.All of the Pis would then function as a single system and provide output rather than providing individual outputs.

The basic working flowchart of the model would be the following:

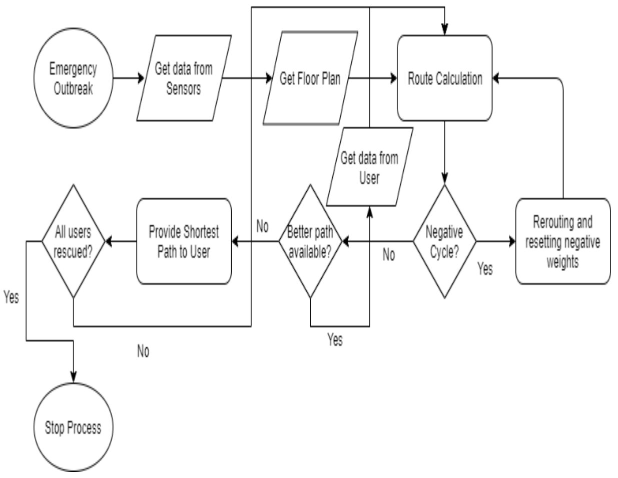
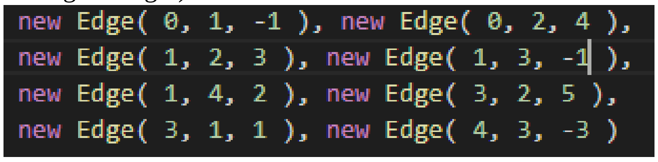


Fig 2:Flow of control during evacuation

As soon as the emergency outbreak occurs,the modules start and the environment plan is generated . After the generation of the graph,the sensors would send metric data from all the surroundings to the algorithm. It would then start to calculate the shortest route based on the weights assigned. If there happens to be a negative cycle in the graph, the weights are reverted to the last change. However if there is a better path available due to the addition of a new weight then it is immediately relayed to the users. This process is repeated till all the users are safely evacuated out of the building.

VII. RESULTS AND DISCUSSION

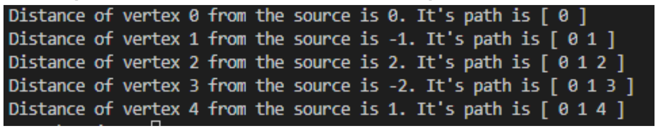
A program was designed to simulate the working of the model. The algorithm was written in Java and made to work by using the inputs from a graph and the weights of the edges. However, the weights of the graphs are fed to the model by the sensors based on the level of fire or smoke or water in the particular room. In the program some negative weights along with high value weights were provided to the program input already to simulate a real time evacuation scenario. In the graph the weights are inserted in the following manner: (X,Y,Z) (Edge from X node to Y node having Z weight).



```
new Edge( 0, 1, -1 ), new Edge( 0, 2, 4 ),
new Edge( 1, 2, 3 ), new Edge( 1, 3, -1 ),
new Edge( 1, 4, 2 ), new Edge( 3, 2, 5 ),
new Edge( 3, 1, 1 ), new Edge( 4, 3, -3 )
```

Fig 3: Providing artificial edge values to algorithm

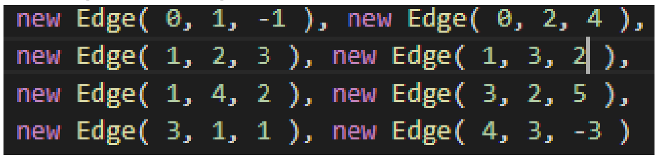
For these edges the output of the program from the starting node(0) to each of the nodes is given as:



```
Distance of vertex 0 from the source is 0. It's path is [ 0 ]
Distance of vertex 1 from the source is -1. It's path is [ 0 1 ]
Distance of vertex 2 from the source is 2. It's path is [ 0 1 2 ]
Distance of vertex 3 from the source is -2. It's path is [ 0 1 3 ]
Distance of vertex 4 from the source is 1. It's path is [ 0 1 4 ]
```

Fig 4: Shortest paths from source to each node

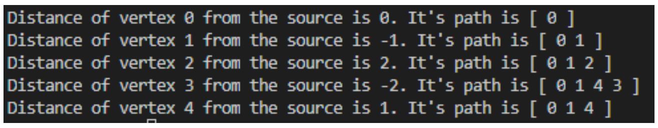
However in simulation of a real life scenario if the 4rth room is under some emergency conditions the value of the weight of the edge is increased from -1 to 2.



```
new Edge( 0, 1, -1 ), new Edge( 0, 2, 4 ),
new Edge( 1, 2, 3 ), new Edge( 1, 3, 2 ),
new Edge( 1, 4, 2 ), new Edge( 3, 2, 5 ),
new Edge( 3, 1, 1 ), new Edge( 4, 3, -3 )
```

Fig 5: Updated value of weight

Then the algorithm changes its paths according to the new weights and returns new shortest routes to every other node starting from source node (0).



```
Distance of vertex 0 from the source is 0. It's path is [ 0 ]
Distance of vertex 1 from the source is -1. It's path is [ 0 1 ]
Distance of vertex 2 from the source is 2. It's path is [ 0 1 2 ]
Distance of vertex 3 from the source is -2. It's path is [ 0 1 4 3 ]
Distance of vertex 4 from the source is 1. It's path is [ 0 1 4 ]
```

Fig 6: The shortest paths are updated

Thus we observe that the simulation is correctly showing the details of the shortest path from the source node 0 to each and every node. However as the graph size increases we observe a considerable shift in the time curve of Bellman Ford

Algorithm as it goes through all the nodes checking for a negative cycle.

Dijkstra's is observed to be faster and more efficient but in the case of negative weights it fails to detect the presence of a negative cycle in the graph. Since this algorithm is primarily focussed on implementing negative weight scenarios in emergency situations, we cannot use Dijkstra's Algorithm. However, a simple comparison of running time vs Graph size is given below for reference.

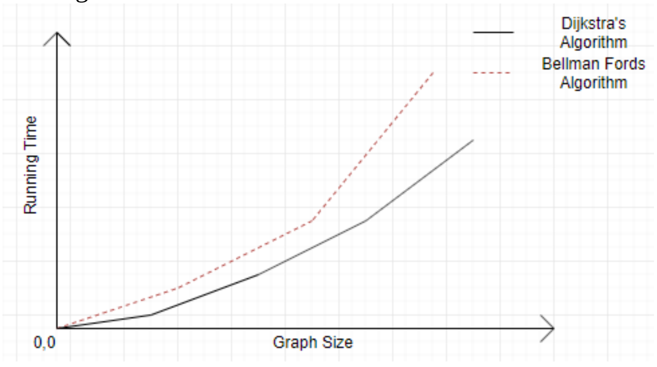


Fig 7: Comparison of running times of two algorithms

The program can now be tweaked to get input from sensors and modify weights accordingly. The output is now traversable and can be transmitted to local devices through a transmitter integrated on a Raspberry Pi.

VIII. CONCLUSION:

Concluding on terms of the algorithm, we can say that we designed a scalable and efficient evacuation plan of any building by designing it with a computational module and then significantly decreasing the amount of bottlenecking and panic created in emergency situations by implementing the possibility of negative weights in the graph.

The idea is to integrate sensors and cameras in order to monitor the fire/flood severity in each node and get feedback through the users during evacuation which is completely dynamic in nature.

Allowing negative weights in the graph during an emergency condition could help in faster evacuation if done in the correct manner. The only problem lies in the fact that negative weights in an evacuation model does not have a hard and fast rule. Any kind of conditions can be implemented as negative weights based on the kind of emergency situation and the location of the emergency. It can also depend on adversities faced during an emergency which is different for different situations.

The algorithm is slow when compared to the Dijsktra's Algorithm because it scans each and every node and relaxes/places values for the edges at the same time. So, for large buildings the time complexity of the algorithm would be a significant challenge during the outbreak of an emergency.

FUTURE WORK:

Some of the future work which can be done to improve this model would be:

1. Integration of artificially intelligent lighting which would greatly aid during evacuation as during panic situations, mobile devices are far less likely to be used when compared to emergency lighting coupled with the model
2. Use of advanced computational power to fuel the model as Bellman Ford's is a dynamic approach and is slower than other models, such as Dijkstra's.
3. Use of cameras which automatically detect fire and water levels so that a user wouldn't have to go through the nodes to figure them out.
4. Improving the algorithm to perform better and faster when a large weighted graph is given as an input.

REFERENCES:

1. Mobile Fire Evacuation System for Large Public Buildings Based on Artificial
2. Intelligence and IoT,Huxian Jiang
3. An Analysis of Least-Cost Routing using Bellman Ford and Dijkstra Algorithms in Wireless Routing Network,Mohd Azlishah Othman, Hamzah Asyrani Sulaiman, Mohd Muzafar Ismail, Mohamad Harris
4. Misran, Maizatul Alice Binti Meor Said, Ridza Azri Ramlee
5. Building Fire Rescue with Evacuation Management Information System and its application
6. XU Tao, MAO Guozhu*, LI Xin, ZHAO Lin School of Environment Science and Technology, Tianjin University, Tianjin 300072, China
7. An Evacuation Algorithm for Large Buildings*Xingxuan Wang and Huan LiuDepartment of Electronic Engineering, Fudan University
8. Handan Rd. 220, Shanghai 200433,

AUTHOR'S PROFILE

Baby D Dayana received her M.E. in Computer Science and Engineering in Government College of Engineering, Tirunelveli. She is currently an Assistant Professor at SRM Institute of Science and Technology since 2017.

Shraman Das, is a Computer Science Engineering student at SRM Institute of Science & Technology,Ramapuram,Chennai.

Shiv Pratap Singh, is a Computer Science Engineering student at SRM Institute of Science & Technology, Ramapuram,Chennai.

Pankaj Gautam, is a Computer Science Engineering student at SRM Institute of Science and Technology,Ramapuram,Chennai.