Software Engineering | Batch 2022

Data Structures and Algorithms
SIT221
Semester 3, 2023-24

Smart Evacuation App Proposal

Submitted By:
Name: Anneshu Nag
Roll No: 2210994760

Submitted To:
Dr. Saravjeet Singh
Unit Chair

Table of Contents

Purpose of the Proposal

The main purpose of this application proposal is to create a strategic plan for the development of a Smart Evacuation app for the rural town of Buninyong, which can also be adopted for other rural areas. Overall, the proposal aims to enhance the safety of residents during bushfires by optimizing evacuation routes and ensuring timely evacuations.

Background

The bushfire in Buninyong in 2019-2020 was a devastating event, which forced the need for and extensive and effective evacuation plan. This led the State Government to commit to reducing bushfire risks and enhance the safety of the people through technological solution. The Smart Evacuation app is the result of this commitment to develop and effective solution that helps the people here so that they can mitigate the danger of bushfire due to the lack of roads.

Buninyong is historically significant and has a population of around 4000 people, which can increase during special events. The town faces annual threat of bushfires, and not only that, there are only four single-lane roads, which clearly is not efficient enough for people to evacuate. During bushfires, limited visibility and blocked roads are some big problems. Additionally, local roads within the town can become impassable if a fire enters the town. To solve them a clear evacuation plan is required.

The proposed Smart Evacuation app seeks to address these challenges by leveraging real-time data on fire location, road capacity, and traffic conditions to guide residents safely out of the area. While initially designed for Buninyong, the solution will be adaptable for other regional and rural communities facing similar evacuation challenges.

Solution

The approach to the creation of the Smart Evacuation App is ground in the utilization of network flow algorithms, adapted to handle multiple sources and sinks, dynamic network updates in the event of road blockages or fires, and the real-time guidance of residents at intersections based on flow calculations.

The proposed solution not only optimizes the evacuation process but also ensures that all residents evacuate via the shortest, unblocked, and uncongested paths, prioritizing their safety.

Implementation – Data Structure

Network flow algorithms are operations designed to model and optimize the flow of resources through networks (*DAA | Flow Networks and Flows - Javatpoint*, n.d.). A network consists of a node and edges. Each edge connect a pair of nodes. Each edge has a capacity that restricts the amount it can transfer *(Hoppe, 1995).*

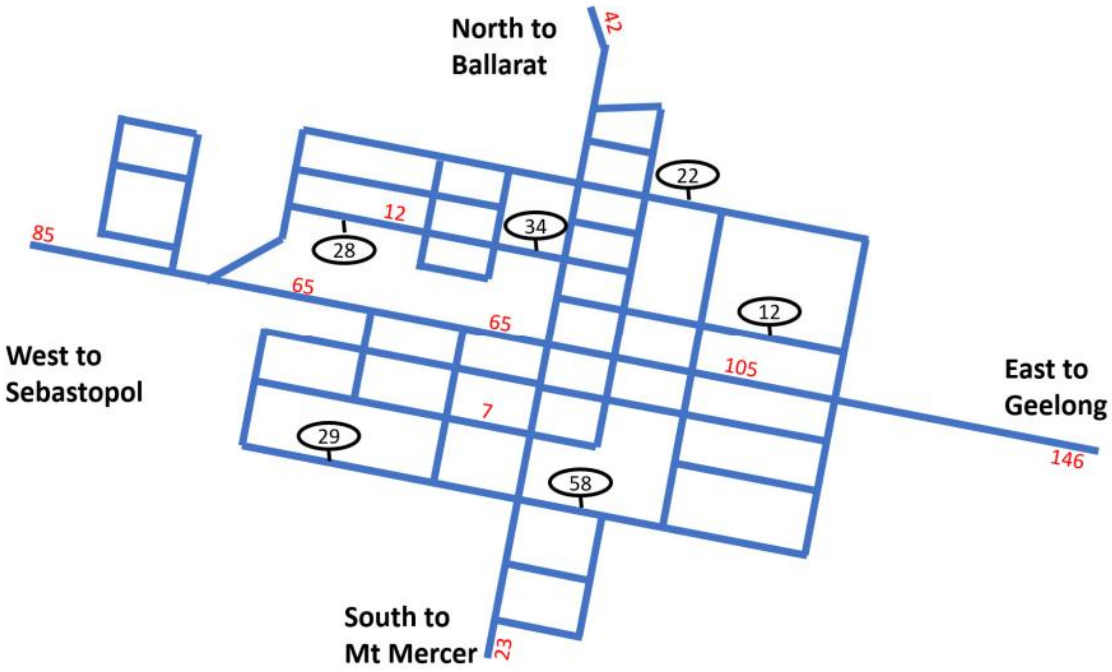


Fig [1]: Simplified map of Buninyong showing the major roads and the exit points at North, South, East and West. Black Circles represents the group of residents (vehicles) living between two intersections. The red numbers show maximum amounts of vehicles that can travel safely on that road.

The road network of Buninyong is considered as a directed graph, where nodes represent intersections or road segments, and edges represent the roads themselves. Each edge is associated with a capacity value, indicating the maximum number of residents that can pass through it in a given time period (e.g., 5 minutes). This capacity value accounts for factors such as road width, terrain, and roadblocks.

In the graph-based model, we include nodes representing the starting locations of residents (sources) and nodes representing the evacuation points (sinks). These nodes are interconnected with the road network through edges that denote potential evacuation routes. The model includes information about the number of residents at each source and the time required to reach each evacuation point from various starting locations.

In the proposed solution, this model includes details of road capacities (total regardless of direction), resident starting locations, and evacuation points. Each road segment between intersections will have its capacity, and each section of road will contain information about resident groups living in that area.

Algorithm Analysis

Let's examine how the existing solutions now address these issues before delving into the specifics of the algorithm implementation of the system.

The majority of the current evacuation systems employ the Djikstra algorithm or some of its components to determine the shortest path, whilst the Edmon-Karp algorithm is frequently used to determine the network's maximum flow. An innovative strategy is needed to create an evacuation system that not only accounts for the shortest path but also provides maximum evacuation for a continually changing network with numerous sources and sinks.

Djikstra Algorithm & Its Shortcomings

The Djikstra algorithm finds the graph's shortest path between any two vertices *(Ahuja, Magnanti, & Orlin, 1988)*. However, it doesn't take into consideration the network (road) alterations that are typical in a wildfire scenario, such as if certain roads are blocked due to various reasons. Additionally, it ignores capacity restrictions, which is significant *(Deng, Chen, Zhang, & Mahadevan, 2012)*. Lastly, a good evacuation plan should have several sinks (exit routes), which the Djikstra algorithm generally doesn't account for.

Pseudo-Code for Djikstra Algorithm *(Abba, 2022)*

```
function Dijkstra(Graph, source):

for each vertex v in Graph.Vertices:
dist[v] ← INFINITY
prev[v] ← UNDEFINED
add v to Q
dist[source] ← 0


while Q is not empty:
u ← vertex in Q with min dist[u]
remove u from Q

for each neighbor v of u still in Q:
alt ← dist[u] + Graph.Edges(u, v)
if alt < dist[v]:
dist[v] ← alt
prev[v] ← u
return dist[], prev[]
```

Time Complexity of Djikstra Algorithm

O (|E| log |V|) is the time complexity for each resident's route calculation, where |E| is the number of edges and |V| is the number of vertices in the road network.

Edmonds-Karp Algorithm & Its Shortcomings

The Edmonds-Karp algorithm is employed to determine the maximum flow that can be transferred from a source to a destination within a network. It operates through an iterative breadth-first search (BFS) technique, which is particularly valuable for assessing the capacity of a path in the network to evacuate people *(Mallick, Khan, Ahmed, Arefin, & Uddin, 2016)*.

However, this algorithm concludes its operation when no further capacity-enhancing paths exist from the source to the destination. In emergency situations where the goal is to evacuate all individuals, even if the maximum flow has already been achieved, this termination condition may be inadequate. Additionally, it is worth noting that similar to the Dijkstra algorithm, Edmonds-Karp is not designed to handle scenarios with multiple sets of source and destination pairs.

Pseudo-Code for Edmonds-Karp Algorithm *(What Is the Edmonds-Karp Algorithm?, n.d.)*

```
inputs:

    C[n x n] : Capacity Matrix
    E[n x n] : Adjacency Matrix
    s : source
    t : sink

output:

    f : maximum flow

Edmonds-Karp:

f = 0;
res_graph = net_graph

while res_graph contains an s – t path P do:

     Suppose P be an s – t path in the residual_graph with of edges.
     P = Breadth-First-Search(C, E, s, t, F)

     Augment maximum_flow using P.
     u = P[v]
     F[u, v] = F[u, v] - m

     Update residual_graph
     F[v, u] = F[v, u] + m
     v = u

end while
return maximum_flow
```

Time Complexity of Edmonds-Karp Algorithm

$O(V.E^2)$ is the time complexity for this algorithm where E is the number of edges representing the road segments and V is the number of vertices representing the residents, evacuation points.

Proposed Algorithm Design

The algorithm's design must take into consideration a wide range of potential situations, including factors such as changing road conditions, the possibility of roads reaching their maximum capacity, and any other unforeseen issues that may arise. Therefore, the proposed algorithm must adopt a different approach that integrates the advantages of the previously mentioned algorithms while mitigating their drawbacks.

Required Modifications for Proposed Algorithm

1. **Multiple Sources and Sinks**: The algorithm should have the capability to accommodate multiple starting points (representing residents) and multiple destinations (representing evacuation locations), effectively directing residents to the closest available evacuation point.

2. **Finding Alternative Networks:** In case of road closures, the algorithm will need to find alternative network to find other open routes for residents to evacuate. If a road is closed, the algorithm will update paths for the evacuees in real-time, ensuring they are directed to the nearest open road.

3. **Terminating Flow from Evacuated Source:** The algorithm will terminate the flow from a source once all residents from that source have safely evacuated for efficient algorithm.

4. **Directing Vehicles at Intersections:** The algorithm will provide directions at intersections based on the number suggested by the flow algorithm. For example, if the algorithm directs 20 residents to the left and 5 to the right, the app will instruct 20 cars to go left and 5 to go right during each time-period.

5. **Accounting for Dynamic Changes in the Path (road):** In bushfire scenarios, the path is not necessarily fixed because there may be instances where the fire has advanced near to the town and is encroaching the roads. The algorithm must take into consideration these dynamic changes in the path and redirect the path as necessary to ensure safe and efficient evacuations.

Pseudo Code of the Proposed Algorithm *(Lu, George, & Shekhar, 2005)*

```
procedure SmartEvacuationAlgorithm(graph, residents, evacuationPoints)
    // Input:
    // - graph: The road network graph with edge capacities and travel times.
    // - residents: List of residents with their current locations.
    // - evacuationPoints: List of evacuation points (destinations).

    // Create a flow network based on the town's road network
    initializeFlowNetwork(graph)
```

```
    // Add a super source and super sink
    addSuperSourceAndSink(graph, residents, evacuationPoints)
    evacuationPlan = {}  // Initialize an empty evacuation plan

    while evacuationNotComplete(residents) do
        // Check for road blockages or fires and update edge weights
        updateRoadConditions(graph)

        // Find shortest, unblocked, uncongested paths using Dijkstra's algorithm
        for each resident in residents do
            source = resident.location
            // Choose the nearest evacuation point
            destination = selectEvacuationPoint(evacuationPoints)
            route, routeDistance = dijkstra(graph, source, destination)

            if route is not empty then
                // Calculate the maximum flow for the selected route
                maxFlow, graph = edmondsKarp(graph, source, destination, route)
                schedule = calculateEvacuationSchedule(route, maxFlow)

                // Update the evacuation plan
                evacuationPlan[resident] = (route, schedule)
                // Move the resident along the path
                updateResidentLocation(resident, route, maxFlow)
            end if
        end for

        // Update estimated evacuation time for central command
        timeEstimate(evacuationPlan)
    end while

     // Output:
    // - evacuationPlan: A mapping of residents to evacuation routes & schedules.
    return evacuationPlan
end procedure

function addSuperSourceAndSink(graph, residents, evacuationPoints)
    // Add a super source and super sink to the flow network
end function

function dijkstra(graph, start, end)
    // Implement Dijkstra's algorithm to find the shortest path
    // Return the shortest route and distance
end function

function edmondsKarp(graph, source, destination, route)
    // Implement the Edmonds-Karp algorithm to find maximum flow on given route
    // Return the maximum flow and update the flow network in graph
```

```
end function

function calculateEvacuationSchedule(route, maxFlow)
    // Calculate the schedule for evacuees along the route based on maximum flow
    // Return a schedule indicating when each evacuee should move
end function

function timeEstimate(evacuationPlan)
    // Update central command with estimates for full evacuation time
end function
```

Mitigating the Problem of Multiple Sources and Sinks

The algorithm addresses the challenge of multiple sources and sinks by combining the multiple sources and sinks into a single source and sink with capacity equal to that of the total capacity of all the sources and sinks respectively *(Lu, George, & Shekhar, 2005).* The algorithm then uses the Edmonds-Karp algorithm on this modified flow network to calculate the maximum flow.

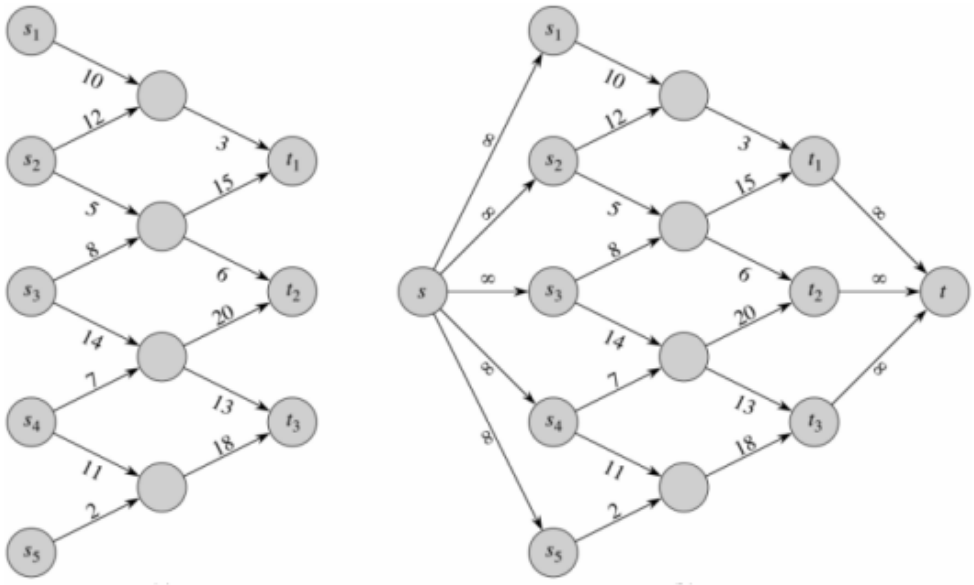


Fig [2A]: Multiple Sources and Sinks; Fig [2B]: Combining Multiple Sources and Sinks into Super Source and Super Sink

Efficiency of the Proposed Solution

The proposed solution offers several advantages in terms of time and space efficiency:

1. **Time Complexity:**
   The algorithm's time complexity is primarily dependent on the two algorithms that have been incorporated to create the evacuation algorithm. For the worst case, let us assume that the algorithm iterated over each person once for say ‘n’ evacuees.

Hence, the worst case complexity for the proposed algorithm will be **O (N * (|E| + |V| log |V| + |E|² * |V|)).**

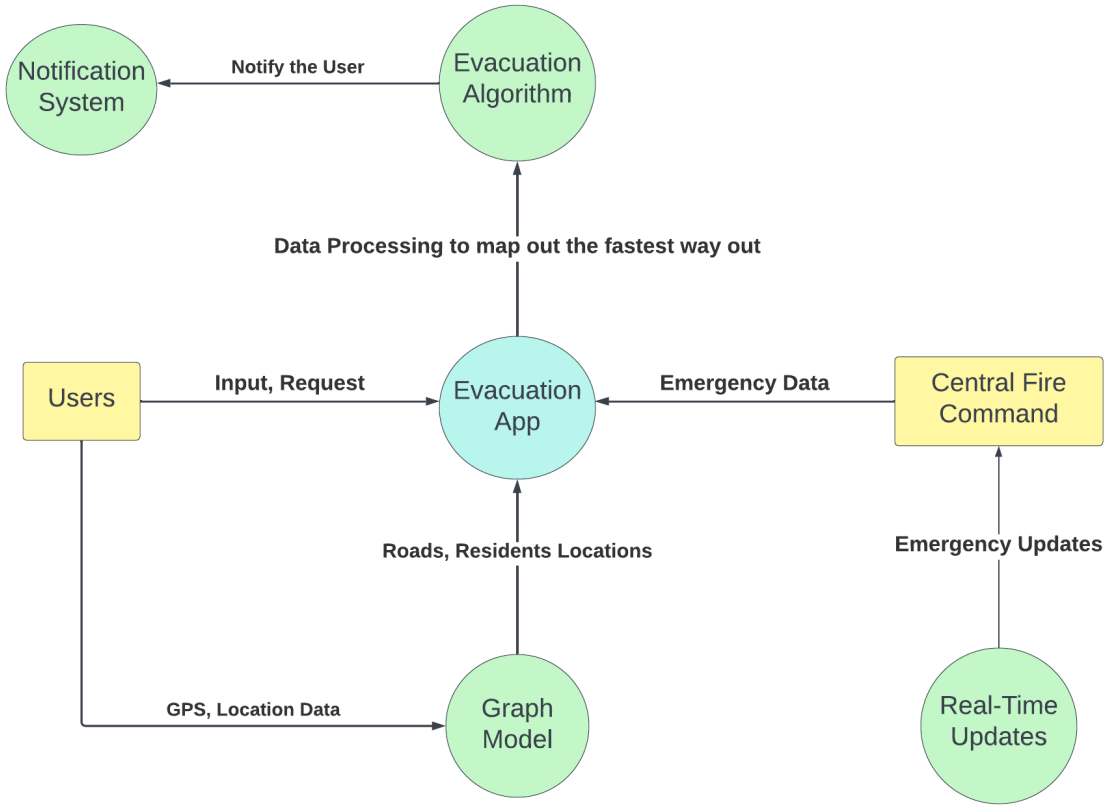
2. **Space Complexity:**

The space complexity primarily depends on data structures used for representing the road network, flow network, residents, and intermediate variables. The space complexity of Djikstra's algorithm is **O (|V| + |E|)** and the space complexity of Edmonds-Karp algorithm is **O (|V|²)**.

Assuming 'N' be the number of evacuees and 'M' be the variable that will represent the evacuation plan given by the algorithm. Hence, the overall space complexity for the algorithm will be **O (|V| + |E| + |V|² + N * M)**.

Data Flow Diagram for the Proposed System

SMART EVACUATION SYSTEM DATA FLOW DIAGRAM



Made by Anneshu Nag, Student ID- 2210994760

Conclusion:

In conclusion, the Smart Evacuation app for Buninyong is critical to enhance the safety of residents during bushfires. The proposed solution, based on network flow algorithms with necessary modifications, offers an efficient and adaptable approach. It addresses the unique challenges of handling multiple sources and sinks, reevaluating the network during road closures, and directing vehicles at intersections.

The app's software will acquire GPS data from the user's phones based on their location. Then it will create a direct weighted graph by overlaying it with a map of the local city's road network. The multiple sources and sinks of evacuation locations will be integrated into a single super source and sink and finally the evacuation algorithm will be implement to route the shortest path of evacuation for the people in the fastest possible time.

By optimizing the evacuation process, the app will contribute to saving lives and protecting the community in Buninyong and can be easily adapted for other regional and rural communities facing similar challenges.

This proposal sets the foundation for an effective Smart Evacuation app, aligning with the State Government's goal of using technology to reduce bushfire risks and improve resident safety.

References:

DAA | Flow Networks and Flows - javatpoint. (n.d.). www.javatpoint.com.

https://www.javatpoint.com/daa-flow-networks-and-flows

Hoppe, B. E. (1995). Efficient dynamic network flow algorithms. Cornell University.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1988). Network flows.

Deng, Y., Chen, Y., Zhang, Y., & Mahadevan, S. (2012). Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. Applied Soft Computing, 12(3), 1231-1237.

Abba, I. V. (2022, December 1). Dijkstra's Algorithm – Explained with a Pseudocode Example. freeCodeCamp.org.

https://www.freecodecamp.org/news/dijkstras-algorithm-explained-with-a-pseudocode-example/

Mallick, K. K., Khan, A. R., Ahmed, M. M., Arefin, M. S., & Uddin, M. S. (2016). Modified EDMONDS-KARP algorithm to solve maximum flow problems. Open Journal of Applied Sciences, 6(2), 131-140.

What is the Edmonds-Karp algorithm? (n.d.). Educative.

https://www.educative.io/answers/what-is-the-edmonds-karp-algorithm

Lu, Q., George, B., & Shekhar, S. (2005, August). Capacity constrained routing algorithms for evacuation planning: A summary of results. In International symposium on spatial and temporal databases (pp. 291-307). Berlin, Heidelberg: Springer Berlin Heidelberg.