Accurate Evacuation Route Planning Using Forward-Backward Shortest Paths

Nishaben Patel
MS Graduate of CS
South Dakota State University
Brookings, SD 57007
Email: nisha04@gmail.com

Manki Min
Dept of EECS
South Dakota State University
Brookings, SD 57007
Email: manki.min@sdstate.edu

Sunho Lim
T^{2}WISTOR, Dept of CS
Texas Tech University
Lubbock, TX 79409
Email: sunho.lim@ttu.edu

***Abstract*—We are living in the 21st century in which technological advances have greatly improved the quality of human lives. However nature keeps presenting its challenges in the form of tsunamis, floods, wild fires, hurricanes, volcanoes, etc. Excellent evacuation route planning is of paramount importance to reduce loss of lives during such disasters. The parameters of finite time lined capacity of evacuation paths and thousands to millions of evacuees for the problem of evacuation route planning makes it more complex than the problem of finding the shortest path which has been studied extensively in graph theory.**

Computation time (execution time hereafter) and evacuation time are two key properties which determine the efficiency and effectiveness of evacuation routing algorithms. This paper proposes a graph theory based evacuation routing algorithm called Forward Backward Shortest Path (FBSP). FBSP uses a novel idea to determine multiple routes by combining multiple paths at a time unlike other evacuation routing algorithms such as Shortest Multiple Path (SMP) that uses just the shortest paths between the source nodes and destination nodes. The experimental analysis on simulated graphs showed that FBSP improves evacuation time over SMP without significantly increasing execution time.

***Index Terms*—Evacuation Routing, Forward-Backward Shortest Paths, Accuracy, Scalability**

I. Introduction

Evacuation of people during the time of disaster is very critical. On March 11, 2011 Japan declared an emergency due to the Fukushima Daiichi nuclear disaster [12] which was caused by earthquake and tsunami. People within a radius of 12 miles were asked to evacuate and about 200,000 people were evacuated. The evacuation process carried out by the government was unorganized and did cause distress to the people [7]. Without careful evacuation plans, such situations can lead to havoc and serious loss of life. In September 2005 Hurricane Rita [19] caused evacuation of approximately 3 million people in Texas, USA. The evacuation routes were spread among multiple cities including Houston, Dallas, Austin, San Antonio and many others. The evacuation paths had long travel times of about 16–36 hours. Within several hours from the beginning of the evacuation, the freeways were full of cars moving at very low pace. The situation demonstrated a lack of efficient evacuation route planning. A study in [3] shows a marked increase in the number and intensity of hurricanes in recent decades. In the event of a terrorist attack [8], [4], safe and timely evacuation of people is vital as well. Without efficient evacuation routing plans, we would continue to face severe hardships as described in the examples above.

We define a transportation network as a graph G, having a set N of nodes and a set E of edges. Edges connect the nodes in the graph and are grouped together to form a path between nodes. Each edge has two important properties, edge capacity and travel time. The capacity of the edge is the maximum number of evacuees which can travel through the edge at a discrete unit of time. Travel time of an edge is the time taken for an evacuee to reach from the source node of the edge to its destination node. Each node has two properties, initial node occupancy and node capacity. Initial node occupancy is the number of evacuees who are present at the node at the beginning of the evacuation. The objective of the problem is to calculate the minimum evacuation time, i.e. the time when the last evacuee arrives at a destination node. Along with the evacuation time, the entire evacuation schedule is expected to be output.

Our objective is to provide a scalable algorithm which has shorter evacuation time than existing algorithms. Our contribution towards this effort is Forward Backward Shortest Path (FBSP) evacuation routing algorithm. FBSP models the evacuation area as a graph with evacuation places as source nodes, shelter places as destination nodes, roads between places as edges, and evacuation routes as paths.

Prior approaches have focused on finding only the shortest paths available between the source and destination nodes and those approaches share the following difficulties:

- Due to the large number of evacuees (at least thousands) and the limited capacity of the paths in evacuation routing problem, the evacuees would need to be sent in batch based on the available capacity of the path.
- Different batches may need to be sent at different start time on the same path as well.

The remainder of this document is organized as follows. Section II discusses the related work carried out for the evacuation route planning problem in literature including CCRP++ and SMP. In Section III we present our proposed approach and describe our FBSP algorithm. Section IV provides the experimental results of the comparison performed for the three algorithms: CCRP++, SMP and FBSP. Section V states our conclusive remarks and future work.

II. Related Work

Existing evacuation route planning can be divided into mainly two categories: linear programming methods and heuristic models. Evacuation routing algorithms can also be used to solve contraflow or lane reversal problems [11], [17]. Polynomial-time approximation scheme (PTAS) for evacuation routing was studied in [9] as well.

A. Linear Programming

Linear programming models the graph as a network flow problem. The original network is repeated at each discrete time to form a time expanded graph. It is then solved as a minimum cost network flow problem to provide an optimal evacuation routing plan [1]. Algorithms using linear programming methods are reviewed in [6]. The advantage of linear programming approach is that the evacuation routes provided are optimal. However it is not feasible to apply linear programming approach to large networks because of its exponential running time.

B. Heuristics

The heuristic approaches do not use time-expanded networks. They produce sub-optimal results at a very low computational cost compared to the time-expanded network based techniques. They use a time-marked list to represent the capacity of each edge at each discrete time. SRCCP and MRCCP were one of the earliest routing algorithms which were based on heuristic capacity constraints [13]. These algorithms were followed by Capacity Constrained Route Planner (CCRP) algorithm [14] that has smaller computation time than MRCCP due to the fact that it runs a single search of the shortest path in each iteration, compared to multiple searches in MRCCP. The shortest path calculation is based on Dijkstras algorithm [2]. The use of the time aggregated model [5] in CCRP has advantages over the time expanded network model. Nonetheless CCRP suffers from large execution time due to repeated calculations for large input data making it not scalable [10]. CCRP++ proposed in [20] has smaller computational time than CCRP. The running time of CCRP++ is at least $O(m(n/s)\lg(n/s))$, where s is the number of source nodes in the network. A quickest path based evacuation routing as an alternative to shortest path evacuation routing has been proposed in [15]. Synchronized flow which is a set of flows that can be used together at the same time has been proposed in [16]. Static Multiple Path (SMP) is a sub-optimal heuristic based evacuation routing algorithm [18]. Unlike CCRP++, SMP finds the shortest (and shorter) paths, which are called static paths, before the evacuation process begins. After finding a path, SMP temporarily reduces the capacities of the edges on the path by the path capacity and find the shortest path again.

C. Running Time Analysis

The main portion of the running time analysis of the aforementioned algorithms will be about the Dijkstra's algorithm to find the shortest paths. The earlier naive analyses provided in section II-B rely on the assumption that Dijkstra's algorithm will always have the same running time which practically does not hold. Instead we present an aggregate approach to analyze the total running time from the beginning to the end of each algorithm CCRP++ and SMP with the main focus on Dijkstra's algorithm.

From the breakdown of the running time of Dijkstra's algorithm shows that the major complexity component lies in RELAX operation. CCRP++ runs Dijkstra's algorithm iteratively and each run has different auxiliary data (list of availability of each edge). Hence we need comprehensive scanning of timed capacities of every edge that can be involved in a path from the source node to the node u that is being relaxed. This scanning operation requires at least $O(ni)$ time since we have up to ni timed capacities. This list of timed capacity must be maintained throughout the entire running of CCRP++ and the update of the list requires $O(hi)$, where h is the maximum number of edges in paths. Even though the relax operation and timed capacity update operation seem similar, there are significant differences about the possible number of edges we need to visit (to find its earliest available time): the former may be called for the path with up to $n-1$ edges while the latter will be called for the path with up to h edges. This update of the timed capacity list is required in SMP as well.

Theorem 1. *The running times of CCRP++ and SMP are at least $O(n^2m^2)$ and $O(phm^2)$ respectively, where m is the total number of evacuees in the network, p is the total number of paths SMP finds, and h is the maximum number of edges in the paths.*

Proof. Combining all the above, we get the following running time analyses:

The running time of CCRP++ is at least $\sum_{i=1}^{m} O(n^2 i + hi) = O(n^2m^2 + hm^2) = O(n^2m^2)$. The running time of SMP is at least $\sum_{i=1}^{p} O(n \lg n) + \sum_{i=1}^{m} O(ph2i) = O(pn \lg n + phm^2) = O(phm^2)$. The second term in the running time of SMP accounts for the timed capacity update cost which may involve update for edges in up to p paths. Note that the above analyses do not consider the actual cost of maintaining the list which will depend on the data structure used in the implementation of the algorithms. □

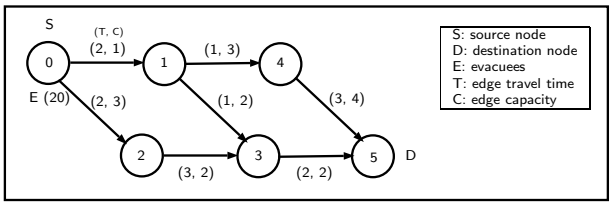
III. Proposed Approach

A. Issues Found

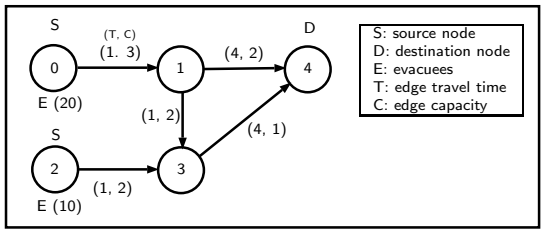
In this section we discuss the three issues uncovered in SMP that we are trying to resolve in our algorithm: 1) Edge-blocking, 2) Evacuation-order-dependency, and 3) Redundant-path-update.

1) Edge-blocking:

One of the shortcomings of SMP is the blocked edges problem which was caused by the way SMP tries to find the paths as described in [18].



(a)



(b)

Fig. 1. Example network graphs

Consider the network in Fig. 1 (a). SMP first finds the path 0-1-3-5 with capacity 1. Then, SMP temporarily reduces the capacity of the edges 0-1 and 3-5 by the path capacity of 1. So the remaining capacities of edges 0-1 and 3-5 are 0 and 1 respectively. SMP then finds path 0-2-3-5 since it cannot find the path 0-1-4-5 because of the zero capacity of the edge 0-1.

We propose to combine forward shortest paths and backward shortest paths to increase the number of paths that we can find.

2) Evacuation-order-dependency:

In SMP the order of evacuation is based on EA which is the end time of a single usage of an evacuation path rather than the time required to evacuate all the people using that path. Consider the graph in Fig. 1 (b). If we evacuate source node 0 always using both paths and hence blocking the path from source node 2, the evacuation time may not be minimum.

We propose to use Evacuation Time (ET) which is an estimation of Combined Evacuation Time (CET) [15] instead of EA to decide the next source node to be used for reservation. ET is defined as below:

$$\text{ET} = \text{EA} + \frac{\text{EVAC}}{\text{Cap}} - 1,$$

where EA is Earliest Arrival time of the path, EVAC is the number of evacuees currently at the source node, and Cap is the capacity of the path.

3) Redundant-path-update:

Many evacuation paths share common edges. SMP algorithm updates all the paths that have shared edges with the reserved path. However unnecessary updates will increase the computation time. Only the paths that must be reserved in the next iteration should be considered for update.

B. Algorithm

In this section we introduce FBSP algorithm. The input to the FBSP algorithm is a graph G. The graph consists of a set N of nodes and a set E of edges between the nodes. The algorithm consists of three main subroutines- Initialization, GetEvacuationPaths and ReservationProcess. Fig. 2 shows the major operations of the FBSP algorithm.

```
FBSP Algorithm
Input:
    G(N, E): a graph G where N is the set of nodes, E is the set of edges,
             S is the set of source nodes, and D is the set of destination nodes
    S: set of source nodes
Output: evacuation time and time-marked evacuation routes

Main:
1. exitNum ← 0
2. evcNum ← number of evacuees on source nodes
3. call GetEvacuationPaths(G)
4. call ReservationProcess

GetEvacuationPaths(G)
1. for each node d ∈ D
2.     for each node n ∈ N \ D
3.         Calculate the shortest path from n to d and add into backward path list
4. for each node s ∈ S
5.     for each node n ∈ N \ {s}
6.         Calculate the shortest path from s to n and add into forward path list
7.         Combine forward and backward paths and add into s's path priority
queue Q

PathReservation
1. while exitNum < evacNum
2.     q ← top element of Q with nonzero evacuees
3.     p ← q.minPath
4.     call ReservePath(p)
5.     call UpdateET
6.     call EvaluatePaths

UpdateET
1. for each node q ∈ Q
2.     for each path p ∈ q.paths
3.         if p.isCandidate == TRUE
4.             p.EA ← p.startTime + p.TT
5.             p.ET ← q.evac / p.capacity + p.EA -1
6.     Find the path p with smallest ET
7.     Update EA, ET, and minPath of q with the corresponding values of p

EvaluatePaths
1. for each node q ∈ Q
2.     for each path p ∈ q.path
3.         if p.isCandidate == FALSE and p.TT ≤ q.EA
4.         p.isCandidate ← TRUE
5.         call IncreaseStartTime(p)
6.         call UpdatePathCapacity(p)
7. call UpdateET
```

Fig. 2. Forward Backward Shortest Path Algorithm

TABLE I
RUNNING TIME ANALYSES FOR CCRP++, SMP, AND FBSP

CCRP++	SMP	FBSP
at least $O(n^2m^2)$	at least $O(phm^2)$	at least $O(Phm^2)$

Theorem 2. *The running time of FBSP is at least $O(Phm^2)$, where m is the total number of evacuees in the network, P is the total number of paths FBSP finds, and h is the maximum number of edges in the paths.*

Proof. Similar to the proof for Theorem 1, we can get the following analysis: The running time of FBSP is at least $\sum_{i=1}^{k} O(n \lg n) + \sum_{i=1}^{m} O(Phi) = O(kn \lg n + Phm^2) = O(Phm^2)$, where k is the sum of numbers of source nodes and destination nodes. □ □

Table I shows the running time analyses for the three algorithms CCRP++, SMP, and FBSP. Note that CCRP++'s

TABLE II
SMP EVACUATION PLAN USING TWO PATHS 0-1-3-5 AND 0-2-3-5 IN FIG. 1 (A)

Time	Path	EA	Cap	TT	Evacuees	Evac. Time
0-5	0-1-3-5	5	1	5	19	5
1-6	0-1-3-5	6	1	5	18	6
2-7	0-1-3-5	7	1	5	17	7
0-7	0-2-3-5	7	1	7	16	7
3-8	0-1-3-5	8	1	5	15	8
1-8	0-2-3-5	8	1	7	14	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮
10-15	0-1-3-5	15	1	5	1	15
8-15	0-2-3-5	15	1	7	0	15

TABLE III
FBSP EVACUATION PLAN USING THREE PATHS 0-1-3-5, 0-2-3-5, AND 0-1-4-5 IN FIG. 1 (A)

Time	Path	ET	EA	Cap	TT	Evacuees	Evac. Time
0-7	0-2-3-5	16	7	2	7	18	7
1-8	0-2-3-5	16	8	2	7	16	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
4-11	0-2-3-5	16	11	2	7	10	11
0-5	0-1-3-5	14	5	1	5	9	11
1-6	0-1-3-5	14	6	1	5	8	11
5-12	0-2-3-5	15	12	2	7	6	12
2-8	0-1-4-5	13	8	1	6	5	12
3-9	0-1-4-5	13	9	1	6	4	12
4-10	0-1-4-5	13	10	1	6	3	12
5-11	0-1-4-5	13	11	1	6	2	12
6-13	0-2-3-5	13	13	2	7	0	13

running time relies on the running time of the modified Dijkstra's algorithm while SMP and FBSP's running times rely on the timed capacity update costs.

C. Issues Resolved

In this section we discuss how FBSP resolves the issues found in section III-A.

1) Edge-blocking:

Referring back to Fig. 1 (a), Table II and Table III show the evacuation plans generated by SMP and FBSP respectively. Using only the two paths 0-1-3-5 and 0-2-3-5, SMP has a total evacuation time of 15. In the tables, the path usages with the same repeated pattern are shadowed with the same color.

Instead of repeated calculations of shortest paths, the path finding in FBSP algorithm is split into two phases of forward / backward shortest path finding. FBSP finds the backward shortest paths (0-1, 0-2, 0-1-3, and 0-1-4) and forward shortest paths (4-5, 3-5, 2-3-5, 1-3-5, 0-1-3-5). Then the two list of paths are combined to the paths (0-1-3-5, 0-2-3-5, 0-1-4-5). Evacuation starts with path 0-1-3-5 which has the lowest travel time of 5. However from start time 2, FBSP uses the additionally found path 0-1-4-5. The total evacuation time of FBSP is 13 which is shorter than that of SMP.

2) Evacuation-order-dependency:

Table IV and Table V are the evacuation plan outputs of SMP and FBSP respectively. SMP starts evacuating from source 0 using the two paths 0-1-4 and 0-1-3-4 until all the evacuees in source 0 are moved to destination node 4. Source 2's evacuee is evacuated at time 0 but after that time no more source 2's evacuees are evacuated since the edge 3-4 is not available which was already occupied by the path 0-1-3-4. Once source 0 has no more evacuees, SMP evacuates source 2 using only one path 2-3-4 starting from time 7. As a result, SMP shows unnecessarily increased evacuation time.

On the contrary, FBSP has a different order of evacuation which results in shorter evacuation time of 14. By using ET not EA, FBSP alternates evacuating source 0 and 2 using the two paths 0-1-4 and 2-3-4. In the meantime, the other path 0-1-3-4 has increased its ET and EA which make itself excluded in the evacuation process. This example clearly shows the effectiveness of ET for the path selection in terms of reduced evacuation time.

TABLE IV
SMP EA UPDATES FOR THE THREE PATHS AFTER EACH PATH RESERVATION IN FIG. 1 (B)

Evacuees	0-1-4	0-1-3-4	2-3-4	Used edges
20 / 10[1]	**5**[2]	6	5	0-1 (0)[3], 1-4 (1)
18 / 10	6	6	**5**	2-3 (0), 3-4 (1)
18 / 9	**6**	6	6	0-1 (1), 1-4 (2)
16 / 9	7	**6**	6	0-1 (0), 1-3 (1), 3-4 (2)
15 / 9	**7**	7	7	0-1 (2), 1-4 (3)
13 / 9	8	**7**	7	0-1 (1), 1-3 (2), 3-4 (3)
⋮	⋮	⋮	⋮	⋮
3 / 9	**11**	11	11	0-1 (6), 1-4 (7)
1 / 9	12	**11**	11	0-1 (5), 1-3 (6), 3-4 (7)
0 / 9	12	12	**12**	2-3 (7), 3-4 (8)
0 / 8	12	13	**13**	2-3 (8), 3-4 (9)
⋮	⋮	⋮	⋮	⋮
0 / 1	12	20	**20**	2-3 (15), 3-4 (16)

[1] Remaining evacuees at source 0 / 1 in the first column
[2] Bold-faced cell represents the reserved path.
[3] The number in parentheses is the time when the edge starts to be used.

TABLE V
FBSP ET / EA UPDATES FOR THE THREE PATHS AFTER EACH PATH RESERVATION IN FIG. 1 (B)

Evacuees	0-1-4	0-1-3-4	2-3-4	Used edges
20 / 10[1]	**14 / 5**[2]	25 / 6	14 / 5	0-1 (0)[3], 1-4 (1)
18 / 10	14 / 6	23 / 6	**14 / 5**	2-3 (0), 3-4 (1)
18 / 9	**14 / 6**	23 / 6	14 / 6	0-1 (1), 1-4 (2)
16 / 9	14 / 7	21 / 6	**14 / 6**	2-3 (1), 3-4 (2)
16 / 8	**14 / 7**	22 / 7	14 / 7	0-1 (2), 1-4 (3)
14 / 8	14 / 8	20 / 7	**14 / 7**	2-3 (2), 3-4 (3)
⋮	⋮	⋮	⋮	⋮
2 / 1	**14 / 14**	15 / 14	14 / 14	0-1 (9), 1-4 (10)
0 / 1	13 / 14	13 / 14	**14 / 14**	2-3 (9), 3-4 (10)

[1] Remaining evacuees at source 0 / 1 in the first column
[2] ET / EA in each cell, bold-faced cell represents the reserved path.
[4] The number in parentheses is the time when the edge starts to be used.

3) Redundant-path-update:

In order to reduce the execution time, FBSP maintains a list of paths called the candidate list to be used for reservation. The paths whose travel time is less than or equal to the source node EA are kept in the path list. Since FBSP finds more paths than SMP, updating all the shared edge paths will incur significant increase in the execution time. FBSP updates the path only when it becomes a candidate for reservation.

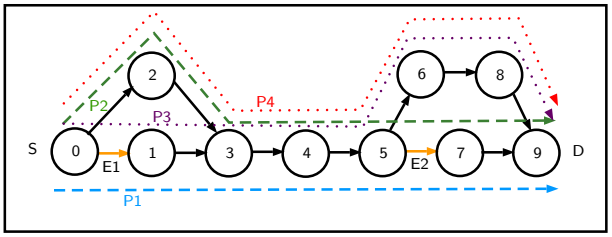


Fig. 3. Shared edge problem in FBSP

4) Remaining Issue:

In some cases, the total evacuation time of FBSP is larger than that of SMP. This issue is caused by the shared edge problem which is described as below.

Consider the graph in Fig. 3. There are four paths P1 (0-1-3-4-5-7-9), P2 (0-2-3-4-5-7-9), P3 (0-1-3-4-5-6-8-9) and P4 (0-2-3-4-5-6-8-9) from the source node 0 to the destination node 9. P1 shares edge E2 (5-7) with P2 and shares E1 (0-1) with P3. Because of these shared edges, paths may not be used for evacuation at the same time. For example, if E1's capacity at a time becomes 0 because of P1, then P3 cannot use E1 at the same time and as a result, P3's EA and ET may be increased. By using a different path 0-2-3-4-5-6-8-9 which cannot be found by FBSP, SMP can find smaller evacuation time. This exact case was found in one of the results.

IV. Simulation Results

We conducted simulation to evaluate our algorithm FBSP in comparison with SMP and CCRP++. The test data was constructed in an $n \times n$ area where n is the number of nodes. The location of a node is randomly generated. A disaster location is selected randomly in the graph. Nodes closest to the disaster location are chosen as source nodes. Nodes farthest from the disaster location are used as destination nodes. The number of source nodes and destination nodes are randomly determined between 1 and 10 and between 1 and 5 respectively. The number of edges generated is proportional to the number of nodes in the graph, with the ratio of the edge to node approximately between 1.5 and 3. Each edge is assigned a random capacity between 1 and 5. The travel time of the edge is proportional to the distance between its end nodes. The graphs are generated so that each source node has at least one path to a destination node. Fifteen graphs for five different node sizes in $\{100, 200, 300, 400, 500\}$ have been created and each source node has randomly chosen number of evacuees. We divided or multiplied each source nodes evacuees by 10 to generate small size or large size graph inputs respectively. A Linux machine with 2.33 GHz dual core CPU and 4GB RAM was used to run the simulation.

A. Evacuation Time Comparison

In this section we compare the evacuation time between FBSP, SMP and CCRP++. Comparison is done using small, medium and large size evacuee groups in the ranges 1,000-10,000, 10,000-100,000, and 100,000-1,000,000. The sizes of the graphs are 100, 200, 300, 400, and 500 nodes. On X-axis we have the number of nodes. Y-axis represents the average normalized evacuation time calculated by

$$\frac{\text{Evacuation time of the algorithm}}{\text{Minimum evacuation time among the algorithms}}.$$

The average normalized value over 15 runs for each type of the graph was used for comparison.

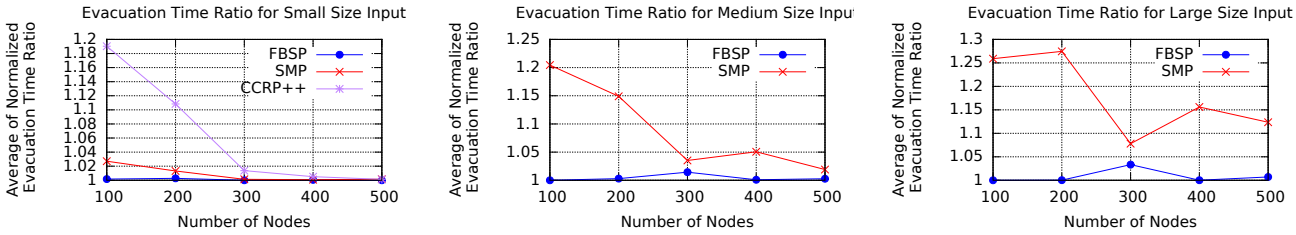
Fig. 4 shows the comparison of the average normalized evacuation time results. In (a) for small size input, both FBSP and SMP perform better than CCRP++. The improvement by FBSP was less for 300, 400, and 500 nodes inputs than for 100 and 200 nodes inputs. The deeper analysis on the path finding and reservation revealed that many of the additional paths found by FBSP had very large travel time and had not been used for evacuation. As a result, FBSP used similar number of evacuation paths as SMP and hence there is no significant improvement by FBSP for small size input. In (b) and (c), CCRP++ took very long time to run for most of medium evacuee graphs and excluded from the comparison. The average normalized evacuation time of FBSP is considerably less than that of SMP, being close to 1 since the evacuation time is increased and long paths found by FBSP can be used together.

FBSP shows better performance in 24%, 58.67%, 65.33% of small, medium, and large input cases, respectively. FBSP and SMP shows same performance in 70.67%, 33,33%, 28% of small, medium, and large input cases, respectively. In any case, only a small portion (up to 8 %) of results show SMP has smaller evacuation time. It is also observed that FBSP outperforms SMP and CCRP++ in terms of evacuation time regardless of the number of evacuees and the size of the graph. In addition, we can confirm that the increased path pool is helpful in reducing the evacuation time especially for large size input.

B. Execution Time Comparison

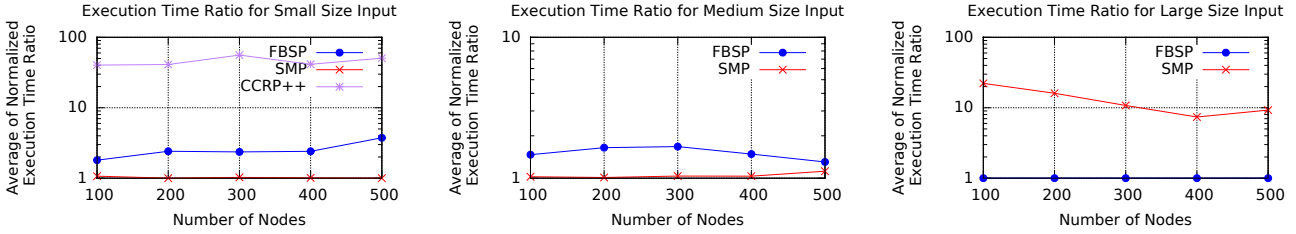
In this section, we compare the execution time between FBSP, SMP and CCRP++ for small, medium and large size evacuee groups. Y-axis represents the average normalized execution time computed similarly as evacuation time.

Fig. 5 shows the execution time comparison. In (a) for small size input, FBSP requires more execution time than SMP due to the more time taken to find and maintain the additional paths but less execution time than CCRP++. In (b), FBSP still requires more time than SMP due to the similar reason as for small size input. However we observe that the difference between the execution times of FBSP and SMP is much smaller (between 1 and 1.5) than the small size input (between 1 and 3.3). In (c) for large size input, FBSP requires smaller execution time than SMP in all cases. The bigger execution time of SMP is due to the linked list implementation used to store the edge capacities at different times. As the number of evacuees grows large, the linked list size keeps increasing. Searching on large linked list increases the overall execution time of SMP.



(a) (b) (c)

Fig. 4. Evacuation time



(a) (b) (c)

Fig. 5. Execution time

V. Conclusion

This study introduces FBSP, a more accurate and scalable algorithm for evacuation routing that can handle large size input of up to millions of evacuees in the network. FBSP resolves the blocked edge issue of SMP where the temporary edge capacity reduction blocks the finding of a useful path. Instead of EA used in CCRP++ and SMP, FBSP uses an enhanced metric ET to determine the order of evacuation, which also considers the number of evacuees and the capacity of the path.

Compared with SMP and CCRP++, FBSP shows more accurate and scalable performance. FBSP is slower than that of SMP in small/medium size input, but faster than CCRP++. This strongly supports scalability of FBSP.

FBSP exhibits a shared edge problem which would be included as one of the future work. Other further efforts would be focused on the reduction of the execution time. One way to achieve it is by removing some of the paths before starting the reservation process. We are planning to perform more thorough analysis of the experimental results as well.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall (1993)

[2] E.W. Dijkstra, A Noye on Two Problems in Connexion with Graph, Numerische Mathematik, vol. 1, pp. 269-271 (1959)

[3] C. Dybas and D. Terraso, Number of Category 4 and 5 Hurricanes Had Doubled Over the Past 35 Years, Press Release 05-162, National Science Foundation, September 15 2005

[4] ESRI, GIS for Homeland Security, An ESRI white paper, November 2001

[5] B. George, S. Kim, and S. Shekhar, Spatio-temporal network databases and routing algorithms: A summary of results, Advances in Spatial and Temporal Databases, Springer Berlin Heidelberg, pp. 460-477 (2007)

[6] H. Hamacher and S. Tjandra, Mathematical Modeling of Evacuation Problems: A State of the Art, Pedestrian and Evacuation Dynamics, pp. 227-266 (2002)

[7] R. Hasegawa, Disaster Evacuation from Japans 2011 Tsunami Disaster and the Fukushima Nuclear Accident, IDDRI, Sciences Po Report 5 pp. 1-54 (2013)

[8] The Homeland Security Council, Planning Scenarios, Executive Summaries, Created for use in National, Federal, State, and Local Homeland Security Preparedness Activities, July 2004

[9] B. Hoppe and É. Tardos, Polynomial time algorithms for some evacuation problems, Proc. ACM-SIAM symposium on discrete algorithms, Society for Industrial and Applied Mathematics, pp. 433441 (1994)

[10] S. Kim, B. George, and S. Shekhar, Evacuation route planning: scalable heuristics, Proc. ACM international symposium on Advances in geographic information systems, pp. 20:1-20:8 (2007)

[11] S. Kim, S. Shekhar, and M. Min, Contraflow transportation network reconfiguration for evacuation route planning, IEEE Transactions on Knowledge and Data Engineering, vol. 20(8), pp. 1115-1129 (2008)

[12] D. Klein and M. Corradini, Fukushima Daiichi: ANS Committee Report, American Nuclear Society (2012)

[13] Q. Lu, Y. Huang, and S. Shekhar, Evacuation planning: a capacity constrained routing approach, Intelligence and Security Informatics, Springer Berlin Heidelberg, pp. 111-125 (2003)

[14] Q. Lu, B. George, and S. Shekhar, Capacity constrained routing algorithms for evacuation planning: A summary of results, Advances in spatial and temporal databases, Springer Berlin Heidelberg, pp. 291-307 (2005)

[15] M. Min and B.C. Neupane, An evacuation planner algorithm in flat time graphs, Proc. ACM International Conference on Ubiquitous Information Management and Communication, pp. 99:1-pp:8 (2011)

[16] M. Min, Synchronized Flow-Based Evacuation Route Planning, Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg. Vol. 7405, pp. 411-422 (2012)

[17] M. Min, Manki and J. Lee, Maximum throughput flow-based contraflow evacuation routing algorithm, Proc. IEEE Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 511-516 (2013)

[18] M. Min, J. Lee, and S. Lim, Effective evacuation route planning algorithms by updating earliest arrival time of multiple paths, Proc. ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS) in conjunction with ACM SIGSPATIAL GIS, (2014)

[19] Hurricane Rita, National Oceanic and Atmospheric Administration. National Climatic Data Center, Updated 22 September 2005

[20] D. Yin, A scalable heuristic for evacuation planning in large road network, Proc. ACM International Workshop on Computational Transportation Science, pp. 19-24 (2009)