Abstract—The evacuation routing problem is NP-hard and hence no polynomial-time algorithm exists for the problem. There have been many studies on the evacuation planning heuristic algorithms but the in-depth comparison between them is not available especially in terms of the performance ratio. Without knowing the heuristic solution quality with respect to the optimal solution, we cannot accurately determine each heuristic algorithm’s strengths and weaknesses that can lead to a better combination of heuristic algorithms. In this paper we present a Linear Programming (LP) based iterative algorithm to assess the evacuation planning algorithms in terms of the evacuation time. The proposed algorithm takes the paths that are found and/or used in a heuristic algorithm as the input and finds the minimum evacuation time using only those paths. To get the optimal solution, our algorithm repeatedly solves relaxed LP formulations formed from the concept of segmented arrival graphs. We segment the arrival graphs of paths based on the edge-sharing and construct the LP formulations along the segmented edges on the graphs. The computational experiment shows that the proposed algorithm computes the solution using time that is comparable to that of the heuristic algorithms. Hence we don’t have an optimal solution to the problem. However the evacuation route planning problem is NP-hard and hence we don’t have an optimal solution to the problem. To work around the shortcomings, we can envision a smart evacuation system in which the disasters are predicted and evacuation routes are planned on-demand and operated automatically. The evacuation route planning algorithm is at the heart of the aforementioned critical system for disaster relief. However the evacuation route planning problem is NP-hard and hence we don’t have an optimal solution to the problem.

In this paper we are proposing an algorithm that computes the minimum amount of time required to finish the evacuation using only the paths, or routes, found by an existing heuristic algorithm. The accurate performance ratio of each heuristic algorithms that we compute by the proposed algorithm will enable us to identify and combine the strengths of each algorithm for better performance. To obtain optimal solutions, we are proposing an LP based approach. By iteratively solving the relaxed LP formulations, we can find the minimum evacuation time. The input to the algorithm consists of the transportation network as a graph and the paths to be used for evacuation. The output of the algorithm consists of the minimum evacuation time and the entire evacuation scenario.

Our contribution in this paper is two-fold:

1) We present an LP formulation based iterative algorithm that computes the minimum evacuation time using only the given paths. The proposed algorithm will be a valuable asset in the evacuation routing study that can provide us the paths-based optimal solution which is typically intractable to obtain.

2) We provide the extensive performance comparison of existing evacuation routing heuristic algorithms using the proposed algorithm. From the comparison we can observe clear advantage and disadvantage of each algorithm using the accurate performance ratio.

The remaining of this paper is composed as follows: Section II summarizes the related work in literature, Section III presents our LP formulation along with our iterative algorithm and analyze the algorithm. In Section IV, we provide extensive computational experiment results showing the comparison between existing evacuation routing algorithms along with the execution time analysis of the proposed algorithm. Finally Section V concludes this paper.

II. RELATED WORK

We can categorize the existing evacuation routing algorithm into LP algorithms and heuristic algorithms. Applications of the evacuation routing algorithms include contraflow evacuation routing problem as in [9], [16]. Theoretical approach such as Polynomial-Time Approximation Scheme was studied in [6]. Nash equilibrium based routing is presented in [10]. In this section, we give summary of algorithms in the aforementioned two categories.
A. Linear Programming

LP models the transportation network using network flows in time-expanded graphs. Time-expanded graphs are the graphs where the original graphs are duplicated at each discrete time between the initial time and the end time both inclusive with proper additional edges between nodes in different times. The advantage of LP approach is that the optimal evacuation routes can be found. However it is not feasible to apply LP approach to large networks because of its exponential running time complexity. A minimum cost network flow problem can be used to achieve an optimal evacuation routing plan [1]. Algorithms using LP methods are reviewed in [5]. A minimum cost Mixed Integer Programming (MIP) formulation is presented in [4] and compared with the presented heuristic algorithm in grid networks. The results show even for the regular grid networks, the MIP algorithm requires running time that grows exponentially. A MIP formulation for general time-expanded graphs is presented in [11] and only the results for the small network size is provided. A multi-objective LP formulation using pre-calculated \( k \)-shortest paths is presented in [19] but the computation was performed on small sized network. A multi-objective multi-commodity flow MIP formulation is presented in [7] and computational results for small sized network is given.

B. Heuristics

In order to reduce the computation time, the heuristic algorithms do not use time-expanded networks. Instead, the heuristic algorithms use timed capacity lists for each edge at each discrete time so that at any discrete time the available capacity of any edge can be tracked properly. Single Route Capacity Constrained Planner (SRCCP) algorithm and Multiple Route Capacity Constrained Planner (MRCCP) algorithm were one of the earliest routing algorithms based on heuristic capacity constraints [12]. A quickest path based evacuation routing as an alternative to shortest path evacuation routing has been proposed in [14]. The estimated evacuation time using a single path instead of the path travel time was used to achieve the better evacuation. Synchronized flow which is a set of flows that can be used together during a period has been proposed in [15]. It models a synchronized flow by grouping disjoint paths from different source nodes.

Capacity Constrained Route Planner (CCRP) algorithm [13] is an improved version of MRCCP with smaller computation time than MRCCP. CCRP runs a single search of the shortest path in each iteration, compared to multiple shortest path searches in MRCCP and hence reduces the running time. In each round CCRP finds the path with the smallest destination arrival time among all pairs of source and destination nodes and evacuates the people along the found path. The destination arrival time is based on the current capacity of each edge and hence it will be different from the mere travel time of a path. The use of this time aggregated model [3] instead of the time expanded network model has advantages such as smaller running time and memory complexity than those of time expanded network algorithms. The shortest path calculation is based on Dijkstra’s algorithm [2]. The running time of CCRP is at least \( O(mn \lg n) \) where \( m \) is the total number of evacuees in the network and \( n \) is the number of nodes but this is a very loose bound. It is shown that CCRP suffers from large execution time due to overly repeated shortest path related calculations for large input data [8].

CCRP++ algorithm based on CCRP was proposed in [20] that further reduces the time for path finding by storing the previous paths. Instead of repeated calculation of the shortest paths for every pair of source/destination, CCRP++ reuses the shortest path found in the previous iterations to reduce the computation time. The previous paths are stored in a priority queue with the key values of the Earliest Arrival time (EA) and the shortest path is only calculated each time only when a path with the EA that is same as the current minimum EA is reserved. The running time of CCRP++ is at least \( O(n^2 m^2) \), where \( n \) is the number of nodes and \( m \) is the number of evacuees in the network [18].

Static Multiple Path (SMP) algorithm is the first evacuation routing algorithm to separate path finding phase from path using phase [17]. The first phase of path finding iteratively searches shortest path from each source node and reduces the capacities of the edges on the found paths so that different shortest paths can be found in the next round. Not only reducing the evacuation time, this approach also reduces the path finding time and hence significantly reduces the total running time as well. The second phase of path using is the actual evacuation process which assign the path with capacity to each evacuee. In each round the source node with the smallest Earliest Arrival time (EA) is selected and the path from the source node with the smallest EA will be assigned to the source node. The running time of SMP is at least \( O(P_S h m^2) \) where \( P_S \) is the total number of paths SMP finds, \( h \) is the maximum number of edges in a path, \( m \) is the number of evacuees in the network [18].

Estimated Evacuation Time (EET) algorithm aims at using more accurate measure to select a path for evacuation in each round [17]. The measure is called EET and means the estimated evacuation time for each source node assuming the maximum possible capacity usage of the currently found paths. This results in more effective routing that finishes earlier than other algorithms due to the enhanced accuracy especially in case of the existence of edge-sharing paths. When multiple paths share an edge, typically not every path can be used at its maximum capacity and hence a single measure such as the travel time does not appropriately reveal the hidden information. Similar to SMP, the running time of EET is at least \( O(P_E h m^2) \) where \( P_E \) is the total number of paths EET finds, \( h \) is the maximum number of edges in a path, \( m \) is the number of evacuees in the network.

Forward Backward Shortest Path (FBSP) algorithm aims at finding more paths for evacuation routing at the same time [18]. As opposed to the previously mentioned algorithm, FBSP has one round of path finding. First backward shortest paths are found from each non-destination node to each destination node. Then forward shortest paths are found from each source node.
node to each non-source node and combine the forward and backward shortest paths to generate a huge path pool that consists of paths with relatively short travel time. The running time of SMP is at least $O(P_F h m^2)$ where $P_F$ is the total number of paths SMP finds, $h$ is the maximum number of edges in a path, $m$ is the number of evacuees in the network [18].

III. PROPOSED ALGORITHM

Fig 1 depicts a simple example graph which has four nodes, 0, 1, 2, 3 and six edges $e_1$, $e_2$, $e_3$, $e_4$, $e_5$, $e_6$. Node 0 is the source node with 30 evacuees and node 3 is the destination node. Every edge has capacity of 1 and the travel time is all 1 except $e_2$ and $e_5$ which have 3 as travel time. From this graph, we can find four possible paths $p_1$, $p_2$, $p_3$, $p_4$. As we can see in Fig 2, we may have different evacuation time depending on the choice of evacuation paths. Fig 2 (a) shows one example of non-optimal choice of evacuation paths which results in the evacuation time of 19 where Fig 2 (b) shows an optimal choice with evacuation time of 18. It is clear that choosing the shortest path for the evacuation does not always result in the minimum evacuation time. The choice of $p_1$, the shortest path, makes $p_4$ with travel time of 4 unavailable and hence $p_3$ with travel time 7 is chosen as in (a). Instead, by choosing short-enough paths $p_2$ and $p_4$ as in (b), both with travel time of 4, we can finish the evacuation earlier than we do in (a).

Difficulties of using the arrival graphs come from the fact that the arrival graphs of paths hide information about the shared edges of paths. Fig 3 shows the hidden information about the edge $e_1$ shared by $p_1$ and $p_2$. Due to the limited capacity of the shared edge $e_1$, it can be used by only one of $p_1$ and $p_2$ at any moment. Hence we may end up with the split use of $e_1$ as in Fig 3 which shows the arrival at the destination node of the edge as well. As a result, the arrival graphs of paths do not properly show us whether the multiple paths are used in maximum capacity or not. Motivated by this observation, we came up with the concept of segmented arrival graphs. Segmenting the arrival graph of a path $p$ means dividing the arrival graph based on time and working on each segment in the combined perspective of all the paths that share an edge with the path $p$. When we divide an arrival graph at a certain time point $t$, we call it as a segmented time mark. This way we can take into consideration every edge-sharing and use it to compute the timed capacity change of edges precisely.

Fig 4 is a slight modification of Fig 1 that can give better explanation of segmented arrival graphs. Assuming that the evacuation time is $t$, we can expect to have arrival graph for every path that ends at $t$ or earlier. Fig. 5 shows the potential arrival graphs for the two paths $p_1$ and $p_2$. The two light grey arrival graphs are for $p_1$ and $p_2$ which share an edge $e_1$ whose arrival graph is represented in dark grey. From the arrival graph of $e_1$, it is clear that from the time mark of 1 till $t-2$, $p_1$ and $p_2$ can share the edge $e_1$. This means that $p_2$’s capacity during the period can be either 0 or 1 and hence we can represent this shared, segmented arrival graph as

$$X_{1,3,t} + X_{2,4,t+1} \leq 2,$$

where $X_{1,3,t}$ represents the amount of flow on the path $p_1$ for the time period of $[3, t]$. Note the different time period for the two paths since the sharing happens at the edge $e_1$ for the time period of $[1, t-2]$.

Before we introduce our relaxed LP formulation, constants and variables are explained.

- $\mathcal{N}$ = set of nodes
- $\mathcal{S}$, $\mathcal{D}$ = set of source and destination nodes, respectively
- $\mathcal{E}$ = set of edges
- $IO_n = $ initial occupancy in node $n \in \mathcal{N}$
- $EC_e, ET_e = $ capacity and travel time of edge $e \in \mathcal{E}$, respectively
- $\mathcal{P} = $ set of paths
- $T_m = $ minimum evacuation time
- $T = $ temporary evacuation time assumed in the algorithm
- $T_{UB} = $ predetermined upper bound of $T$
- $PAT_{p}, EAT_{e} = $ the first time when an evacuee arrives at the destination node of the path $p$ and the edge $e$, respectively
- $\mathcal{P}_s = $ set of paths that starts from the source node $s \in \mathcal{S}$
- $PC_p, PT_p = $ capacity and travel time of the path $p \in \mathcal{P}$, respectively
- $PTM_p, ETM_e = $ array of segmented time marks of the path $p$ and the edge $e$ sorted based on time mark values in increasing order, respectively
- $\mathcal{P}E_e = $ set of paths that pass through the edge $e$
- $PET_{p,e} = \left[ PC_p - EAT_e + ETM_e[l] \right]
- X_{p,t_1,t_2} = $ number of evacuees arrive at a destination node of the path $p$ between the times $t_1$ and $t_2 - 1$

Note that $ETM_e[1] = ET_e$, $PTM_p[1] = PT_p$, and $PET_{p,e}[1]$ can be added to an arrival time of an edge and be transformed into an arrival time of a path.

Fig 6 describes the relaxed LP formulation used in our iterative algorithm. Equation (2) ensures that all the evacuees are in a destination node at time $T$. Equation (3) ensures that at any time the capacity of an edge can be at most the original edge capacity. Equations (4) and (5) represents the bounds of the variables. It is used to test whether a value of $T_{UB}$ is feasible for the given transportation network. The iterative algorithm simply changes the lower bound and upper bound of feasible $T_{UB}$ and eventually finds the optimal solution. Even though the LP formulation itself can be described in
simple form as in the figure, the algorithm to construct such LP formulation is time-consuming. We have tried two different approaches: 1) use unit time marks for both paths and edges, 2) segmenting arrival graphs to find bigger time marks. The trade-offs between the two different approaches is the simplicity of pseudocode and the time-space complexity of the computation. With unit time marks, the pseudocode becomes as simple as the LP formulation in the figure. However the LP size grows quickly, especially with the bigger sized of evacuees and more paths involved in the formulation. Instead, segmenting the arrival graphs results in much smaller sized LP formulation and hence much smaller computation time. The difficulty of the second approach lies in the computation of pseudocode and the time-space complexity of the computation. Proof. Fig 7 segments the arrival graphs of all the paths that are input to the algorithm and creates the corresponding relaxed LP formulation. The segmentation of arrival graphs is essentially finding shared time periods (or time marks) of each edge e by checking when every path is using the edge e and then combining those paths sharing e for each of the shared time periods. Our LP algorithm does the job by repeatedly finding a new time mark for each path and all its edges. The LP formulation will output as feasible if UB is not less than the evacuation time and infeasible otherwise. Fig 8 is a simple binary search algorithm to find a convergence (LB >= UB). Therefore our iterative algorithm will always find the optimal solution.

**Theorem 1.** The proposed iterative algorithm finds the optimal solution, i.e. finds the optimal evacuation time, for using only the provided paths.

**Proof.** Fig 7 segments the arrival graphs of all the paths that are input to the algorithm and creates the corresponding relaxed LP formulation. The segmentation of arrival graphs is essentially finding shared time periods (or time marks) of each edge e by checking when every path is using the edge e and then combining those paths sharing e for each of the shared time periods. Our LP algorithm does the job by repeatedly finding a new time mark for each path and all its edges. The LP formulation will output as feasible if UB is not less than the evacuation time and infeasible otherwise. Fig 8 is a simple binary search algorithm to find a convergence (LB >= UB). Therefore our iterative algorithm will always find the optimal solution.

**Theorem 2.** The proposed iterative algorithm has the running time complexity of $O(T_m \log T_m |\mathcal{P}|^2 |\mathcal{E}|^2)$.

**Proof.** First we analyze line by line the complexity of the LP formulation construction algorithm in Fig. 7.

Lines 1-6 run at complexity of $O(|\mathcal{E}| |\mathcal{P}|)$. Lines 7-9 run at complexity of $O(|\mathcal{P}|)$ since $|ETM_e|$ = $O(|\mathcal{P}_e|)$ and $\bigcup_{e \in \mathcal{E}} \mathcal{P}_e = O(|\mathcal{P}|)$.

Lines 7 and 10 run at complexity of $O(|\mathcal{E}| \log |\mathcal{E}|)$.

Lines 7 and 11-13 run at complexity of $O(|\mathcal{P}|^2)$ since $\sum_{e \in \mathcal{E}} \sum_{t_1, t_2 \in ETM_e} |\mathcal{P}_e| = \sum_{e \in \mathcal{E}} \sum_{t_1, t_2 \in ETM_e} O(|\mathcal{P}|) = O(|\mathcal{P}|^2)$.

Lines 14-16 run at complexity of $O(T_{UB} |\mathcal{P}|^2 |\mathcal{E}|^2)$ since in the worst case every unit time mark may be inserted.

Hence the LP formulation construction requires $O(UB |\mathcal{P}|^2 |\mathcal{E}|^2)$ running time.

Since the range of LB and UB in the algorithm in Fig 8 is first increased to $2T_m$ and then decreased by halves until it converges, we can expect $\log T_m$ repetitions. Each run of the algorithm in Fig 8 runs in $O(UB |\mathcal{P}|^2 |\mathcal{E}|^2) = O(T_m |\mathcal{P}|^2 |\mathcal{E}|^2)$, and hence we have the total running time of $O(T_m \log T_m |\mathcal{P}|^2 |\mathcal{E}|^2)$.

Note that the major component that determines the time complexity of the proposed algorithm is in lines 14-15 which synchronize the time marks for the segmented arrival graphs.
Minimize \( T - 1 \sum_{p \in \Xi, \ 2 \leq i \leq |\text{PTM}_p|} X_{p, \text{PTM}_p[i-1], \text{PTM}_p[i]} = IO_s, \quad \forall s \in \mathcal{S} \) 
(1)
\[
\sum_{p \in \Xi, e} X_{p, \text{PET}_{p,e}[i-1], \text{PET}_{p,e}[i]} \leq E_{C_e} \ast (\text{ETM}_e[i] - \text{ETM}_e[i-1]),
\]
\( \forall 2 \leq i \leq |\text{ETM}_e|, \forall e \in \mathcal{E} \) s.t. \( \sum_{p \in \Xi, e} \text{PC}_p > E_{C_e} \) \( \forall 1 \leq p \leq |\Xi|, \forall 2 \leq i \leq |\text{PTM}_p| \) 
(3)
\( T = T_{UB} \)
\( X_{p, \text{PTM}_p[i-1], \text{PTM}_p[i]} \leq \text{PC}_p \ast (\text{PTM}_p[i] - \text{PTM}_p[i-1]), \quad \forall 1 \leq p \leq |\Xi|, \forall 2 \leq i \leq |\text{PTM}_p| \) 
(4)
\[ \text{PET}_{p,e}[1] \]
(5)

Fig. 6. Relaxed LP formulation to test the feasibility of \( T_{UB} \)

for each path \( p \in \Xi \)
1. for each edge \( e \) on \( p \)
2. Put \( p \) into \( \Xi \) and calculate \( EAT_e \)
3. Calculate \( PC_p \) and \( PAT_p \)
4. for each edge \( e \) on \( p \)
5. Calculate \( \text{PET}_{p,e}[1] \)
6. for each edge \( e \)
7. for each path \( p \in \Xi \)
8. Add \( PT_p - \text{PET}_{p,e}[1] \) and \( T_{UB} - \text{PET}_{p,e}[1] \)
9. into ETM_e
10. Sort ETM_e
11. for each consecutive time mark pair \( t_1, t_2 \in \text{ETM}_e \)
12. if sum of \( PC_p > E_{C_e} \) for all \( p \in \Xi \)
13. Put \( t_1 + \text{PET}_{p,e}[1], t_2 + \text{PET}_{p,e}[1] \)
14. into \( \text{PTM}_p \) for all \( p \in \Xi \)
15. while (new path time mark \( t \) for a path \( p \) and an edge \( e \) is introduced)
16. Construct the relaxed LP (Fig 7) using the generated time marks (segmented arrival graphs)

Fig. 7. LP formulation construction algorithm

\[ LB = 0, UB = 1000 \]
1. while \( LB < UB \)
2. Set \( T_{UB} := UB \) and solve the relaxed LP (Fig 7)
3. if the LP is feasible, \( UB = (UB + LB) / 2 \)
4. else \( LB = UB + 1, UB = UB \ast 2 \)

Fig. 8. The proposed iterative algorithm

IV. COMPUTATIONAL EXPERIMENT

In this section, we provide the computational experiment results comparing the performances of the existing algorithms CCRP++, SMP, EET, and FBSP for small/medium sized input. Small sized input is for 1,000-10,000 evacuees and medium is for 10,000-100,000. For each input size, we generated fifteen input graphs of different sizes of \{100, 200, 300, 400, 500\} each. We first randomly chose each node’s location and selected a disaster location randomly. Then the closest/farthest nodes were chosen as the source/destination nodes, respectively. We chose 1-10/1-5 random source/destination nodes and the number of edges is approximately 1.5-3 times of the number of nodes. Each edge’s capacity was randomly picked between 1 and 5 and travel time is set proportional to the distance between its two end nodes. Each input graph was ensured that each source node has at least one path to a destination node. For medium sized input, each source node’s initial occupancy was randomly determined between 100 and 15,000. For small sized input, we divided each source node’s initial occupancy by 10 and reduce the entire evacuee size by 1/10. For each heuristic algorithm, we ran the proposed iterative LP algorithm to obtain the optimal evacuation time using only those paths found/used by the algorithm. A Linux machine with 2.33 GHz dual core CPU and 4GB RAM was used to run the simulation. The performance ratio is defined as the performance (evacuation time) of each algorithm over the best performance (either optimal performance or the best heuristic performance).

Fig 9 shows the comparison of the algorithms. The left column shows the results for the small sized input and the right shows for the medium sized input. In each plot, left y-axis shows the results using paths found by the algorithms and the right y-axis shows the results using paths used by the algorithms. For small sized input, each algorithm showed almost same results in both the found paths and the used paths. This is because the evacuee size is too small to require more paths, typically having longer travel time, for evacuation. So for small sized input, we can ignore the difference of performance between path finding and path using. One clear observation is that CCRP++ typically provides the worst performance, meaning the paths found/used by CCRP++ could have been used in other ways to further reduce the evacuation time by. The potential improvement, however decreases as the number of the nodes grows which shows that with 500 nodes, each algorithm will achieve already almost optimal path finding and using. Another observation is that CCRP++, SMP, FBSP, and EET show the relative performance in this order, from worst to best, which is as expected considering their absolute performances.

CCRP++ took too much time for medium sized input and hence was completely ignored. Similarly FBSP took too much time due to too many found paths (generally 10 times more, and 176 times at maximum) and hence was ignored in the
Fig. 9. Ratios of evacuation time to optimum, for small sized input (left), for medium sized input (right)
results with found input. We compared SMP and EET for medium sized evacuees with found paths and SMP, EET, and FBSP with used paths for this reason in Fig 10. Unlike the small input size, medium sized input showed different performance ratio for different algorithms. SMP solutions showed performance ratio of 100.5%-102.5% on average for small sized input, but the performance ratio grew to 102%-120% on average for medium sized input. Another interesting observation is that FBSP, despite its superior performance in other cases, showed some peculiarly worse performance than others in 200 nodes run 5 (168%), 300 nodes run 6 (132%), run 8(145%), 500 nodes run 7 (151%), and run 14 (188%). EET, on the other hand, showed much more stable performance ratio in the range of 101.5%-103.7% for the medium sized input.

Fig 11 shows the execution time ratios to the number of paths for both small and medium sized input. In both figures, the left column represents small sized input and the right column medium. Interesting observation in Fig 11 (a) and (c) is that FBSP, EET, and SMP all have decreasing execution time ratio to the found paths and increasing execution time ratio to the used paths for 300-500 node sizes. Fig11 (a) shows that even though a lot more paths were found by FBSP, not all of them has contribution in increasing the running time. This could possibly mean that many of those long paths have no complicated mutual sharing of edges during the short evacuation time period. On the other hand, Fig 11 (b) and (d) show that EET and SMP have decreasing running time ratio to the number of paths for 300-500 node sizes which is a similar phenomenon as having almost optimal solutions by all algorithms for 500 node sizes. As the input size grows, in this case by increasing the number of evacuees, the path usages in each algorithm becomes tight and leaves not much room for improvement and hence the running time may decrease compared to the increased number of paths.

Fig 12 shows the ratios of used paths to found paths by each algorithm. CCRP++ finds one path and uses it right away so its result is excluded from the graph. Both SMP and EET used most (60%-90%) of the found paths. The ratio of used paths to found paths grew both for SMP and EET from small sized input to medium sized input, which is explained clearly by the increased evacuation time which utilizes more paths with possibly longer travel times. Note that small sized input and medium sized input shares the same underlying graph and hence the numbers of the found paths are the same in both input sizes. Only the number of the used paths may differ based on the input size.

V. Conclusion

In this paper we presented a new LP formulation that can be used to determine the feasibility of the upper bound of the evacuation time using only the given paths. This is the first LP formulation that leads to practical results to analyze existing algorithms' performance on general graph settings. We also presented an iterative algorithm based on the relaxed LP formulation to find the optimal evacuation time using the given paths. The algorithm works in low time complexity (comparable to that of the heuristic algorithms). Using the algorithm as a tool, we compared the performance ratios of the existing evacuation routing algorithms.

Further research directions include the simplification of time mark synchronization, identified as the performance bottleneck in Section III. Based on the improved time mark synchronization, we plan to study on the edge-only LP formulation without paths-related variables to compute the evacuation time. From the small evacuees with found paths results, FBSP shows superior performance compared to the other algorithms and hence we can use the path finding phase of FBSP. From the results of both size inputs with used paths, EET showed the superior and stable performance compared to the other algorithms hence we can use the path using phase of EET. We plan to combine the two algorithms FBSP and EET as we suggested and compare their performance ratios to verify our findings.

References

Fig. 11. Averaged ratios of running times to number of paths, small sized evacuees (left), medium sized evacuees (right)

Fig. 12. Averaged ratios of number of used paths to number of found paths, small sized evacuees (left), medium sized evacuees (right)


