Improved Algorithms for the Evacuation Route Planning Problem

Gopinath Mishra[1], Subhra Mazumdar[2], and Arindam Pal[2(✉)]

[1] Advanced Computing and Microelectronics Unit,
Indian Statistical Institute, Kolkata, India
gopianjan117@gmail.com
[2] Innovation Labs, TCS Research, Tata Consultancy Services, Kolkata, India
{subhra.mazumdar,arindam.pal1}@tcs.com

Abstract. Emergency evacuation is the process of movement of people away from the threat or actual occurrence of hazards such as natural disasters, terrorist attacks, fires and bombs. In this paper, we focus on evacuation from a building, but the ideas can be applied to city and region evacuation. We define the problem and show how it can be modeled using graphs. The resulting optimization problem can be formulated as an integer linear program. Though this can be solved exactly, this approach does not scale well for graphs with thousands of nodes and several hundred thousands of edges. This is impractical for large graphs.

We study a special case of this problem, where there is only a single source and a single sink. For this case, we give an improved algorithm *Single Source Single Sink Evacuation Route Planner (SSEP)*, whose evacuation time is always at most that of a famous algorithm *Capacity Constrained Route Planner (CCRP)*, and whose running time is strictly less than that of CCRP. We prove this mathematically and give supporting results by extensive experiments. We also study randomized behavior model of people and give some interesting results.

1 Introduction

Emergency evacuation is the process of movement of people away from the threat or actual occurrence of hazards such as natural disasters, terrorist attacks, fires and bombs. In this paper, we focus on evacuation from a building, though the ideas can be applied to city and region evacuation. We are motivated by the evacuation drill that regularly happens in our company Tata Consultancy Services. We are developing a system SmartEvacTrak [1] for people counting and coarse-level localization for evacuation of large buildings. Safe evacuation of thousands of employees in a timely manner, so that no one is left behind, is a major challenge for the building administrators. Time is the main parameter in our model. The travel time between different areas of the building is part of the input and the evacuation time is the output. In the following discussion, we use {*graph, network*}, {*node, vertex*}, {*edge, arc*}, and {*path, route*} interchangeably.

We have a building along with its floor plan. Employees are present in some portions (rooms) of the building. There are some *exits* on the floor. Every

Z. Lu et al. (Eds.): COCOA 2015, LNCS 9486, pp. 3–19, 2015.
DOI: 10.1007/978-3-319-26626-8_1

corridor has a *capacity*, which is the number of employees that can pass through the corridor per unit time. Every corridor also has a *travel time*, which is the time required to move from the start of the corridor to the end. The goal is to suggest a feasible route for each employee so that he can be guided to an exit. It must be ensured that at any time the number of employees passing through a corridor does not exceed it's capacity.

A complex building does not provide its occupants with all the information required to find the optimal route. In an emergency, people tend to panic and do not always follow the paths suggested by the algorithm. They are not given enough time to establish a cognitive map of the building. To address this issue, we need to model the behavior of people in emergency situations. We have proposed a simple randomized behavior model and analyzed it. The expected evacuation time comes out to be quite good. None of the previous works considered any behavior model of people.

2 Related Work

In this section, we give a summary of different algorithms for the evacuation route planning problem. Skutella [12] has a good survey on the network flows over time problem. The monograph by Hamacher and Tjandra [4] surveys the state of the art on the mathematical modeling of evacuation problems. Both these papers give a good introduction and comprehensive treatment to this topic.

The LP based polynomial time algorithm for evacuation problem by Hoppe and Tardos [5] uses the ellipsoid method and runs in $O(n^6T^6)$ time, where n is the number of nodes in the graph and T is the evacuation *egress time* for the given network. It uses time-expanded graphs for the network, where there are $T+1$ copies of each node. The expression for time complexity shows that it is not scalable even for mid-sized networks. Another disadvantage is that it requires the evacuation egress time (T) *apriori*, which is not easy to estimate. As the time complexity is a function of T, it is not a fully polynomial time algorithm.

One of the earliest algorithms by Lu et al. [8] is Capacity Constrained Route Planner (CCRP). CCRP uses Dijkstra's generalized shortest path algorithm to find shortest paths from any source to any sink, provided that there is enough capacity available on all nodes and edges of the path. An important feature of CCRP is that instead of a single value which does not vary with time, edge capacities and node capacities are modeled as time series (function of time). Here, we need to update edge and node capacities for each time period. The running time of CCRP is $O(p(m+n\log n))$, ($O(pn\log n)$ for sparse graphs, where $m = O(n)$) and space complexity is $O((m+n)T)$ ($O(nT)$ for sparse graphs). Here m and n denotes the number of edges and the number of vertices of the graph respectively, p denotes the number of evacuees, and T denotes the evacuation egress time. As space complexity is always at most the time complexity, the running time of CCRP is implicitly dependent on T. For sparse graphs, $nT \leq pn\log n$, *i.e.*, $T \leq p\log n$. So, for sparse graphs the evacuation egress time is at most $O(p\log n)$. The space complexity of $O(nT)$ and unnecessary expansion of source nodes in each iteration are two main disadvantages of CCRP.

To overcome the unnecessary expansion in each iteration, Yin et al. [14] introduced the CCRP++ algorithm. The main advantage of CCRP++ is that it runs faster than CCRP. But the quality of solution is not good, because availability along a path may change between the times when paths are reserved and when they are actually used.

Min and Neupane [11] introduced the concept of *combined evacuation time* (CET) and *quickest paths*, which considers both transit time and capacity on each path and provides a fair balance between them. Let there be k edge-disjoint paths $\{P_1, P_2, \ldots, P_k\}$ from source node s to sink node t. Then, the combined evacuation time is given by,

$$CET(\{P_1, P_2, \ldots, P_k\}) = \left\lceil \frac{p + \sum_{i=1}^{k} C_i T_i}{\sum_{i=1}^{k} C_i} \right\rceil - 1 \tag{1}$$

where C_i and T_i denotes the capacity and transit time of path P_i respectively, and p denotes the number of evacuees. Time required to evacuate p people via a path P having transit time T and capacity C is $T + \lceil \frac{p}{C} \rceil - 1$. So, P_i is said to be the *quickest path* if and only if $T_i + \left\lceil \frac{p}{C_i} \right\rceil - 1 \leq T_j + \left\lceil \frac{p}{C_j} \right\rceil - 1$, for all $j \in \{1, \ldots, k\} \setminus \{i\}$.

The formula for combined evacuation time not only gives an exact expression for the evacuation time, but it also gives the number of people that will be evacuated on each path. The intuition behind the concept of CET is that paths having lesser arrival time will evacuate more groups. This algorithm is known as QPER (Quickest Path Evacuation Routing). The algorithm finds all edge-disjoint paths between a single source and a single sink and orders them according to the quickest evacuation time (calculated using CET) and adds them one by one. The algorithm is fairly simple. It does not use time-expanded graphs and there is no need to store availability information at each time stamp, as only edge-disjoint paths are considered. But their algorithm is limited to single source and single sink evacuation problems. Besides these, the addition of paths is not consistent, i.e., a path added at some point of time may be removed by the algorithm at a latter point of time, in case removal makes the solution better.

The solutions produced by CCRP++ and QPER do not follow semantics of CCRP, i.e., the solution quality is not better than that of CCRP. Recently Gupta and Sarda [3] have given an algorithm called CCRP*, where evacuation plan is same as that of CCRP and it runs faster in practice. Instead of running Dijkstra's algorithm from scratch in each iteration, they resume it from the previous iteration.

Kim et al. [6] studied the contraflow network configuration problem to minimize the evacuation time. In the *contraflow* problem, the goal is to find a reconfigured network identifying the ideal direction for each edge to minimize the evacuation time, by reallocating the available capacity. They proved that this problem is NP-complete. They designed a greedy heuristic to produce high-quality solutions with significant performance. They also developed a bottleneck

relief heuristic to deal with large numbers of evacuees. They evaluated the proposed approaches both analytically and experimentally using real-world data sets. Min and Lee [10] build on this idea to design a maximum throughput flow-based contraflow evacuation routing algorithm.

Min [9] proposed the idea of *synchronized flow* based evacuation route planning. Synchronized flows replace the use of time-expanded graphs and provides higher scalability in terms of the evacuation time or the number of people evacuated. The computation time only depends on the number of source nodes and the size of the graph.

Dressler et al. [2] uses a network flow based approach to solve this problem. They use two algorithms: one is based on *minimum cost transshipment* and the other is based on *earliest arrival transshipment*. They evaluate these two approaches using a cellular automaton model to simulate the behavior of the evacuees. The minimum cost approach does not consider the distances between evacuees and exits. It may fail if there are exits very far away. Problems also arise if a lot of exits share the same bottleneck edges. The earliest arrival approach uses an optimal flow over time and thus does not suffer from these problems. But the exit assignment computed by the earliest arrival approach may not be optimal.

There are some previous works which considered the behavior of people in an emergency. Løvs [7] proposed different models of finding escape routes in an emergency. Song et al. [13] collect big and heterogeneous data to capture and analyze human emergency mobility following different disasters in Japan. They develop a general model of human emergency mobility using a Hidden Markov Model (HMM) for generating or simulating large amount of human emergency movements following disasters.

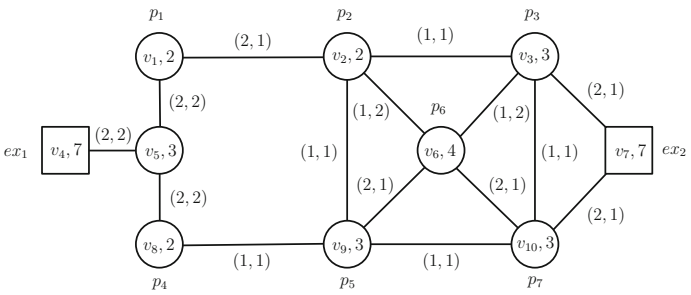

Fig. 1. A building graph, where vertices represented as squares denote exits. The vertex name and capacity are written inside a vertex. The edge capacity and travel time are written beside an edge. Persons residing on a vertex are specified beside that vertex.

3 Problem Definition and Model

The building floor plan can be represented as a graph $G = (V, E)$, where V and E are the set of vertices and edges respectively. The number of vertices and edges are n and m respectively. Nodes represent rooms, lobbies and intersection points and arcs represent corridors, hallways and staircases. Some nodes in the building having significant number of people are modeled as *source* nodes. The exits of a building are represented as *sink* nodes. Each node has a *capacity*, which is the maximum number of people that can stay at that location at any given time and an *occupancy*, which is the number of people currently occupying the location. Here, p is the total number of people who needs to be evacuated.

Each edge has a *capacity*, which is the maximum number of people that can traverse the edge per unit time and a *travel time*, which is the time needed to travel from one node to another along that edge.

Figure 1 shows a building graph that consists of 10 vertices and 15 edges. For each vertex v, it's name and the capacity are specified by a pair of the form $(v, c(v))$. A vertex representing an exit is drawn as a square, while the others are drawn as circles. For each edge e, the capacity and the travel time are specified on the edge by the pair $(c(e), d(e))$. The goal is to find the exit and the path (route) for each employee, subject to the constraint that the number of source-sink paths passing through an edge does not exceed the capacity of the edge at any unit time interval. The objective function we want to minimize is the total time of evacuation, that is the time at which the last employee is evacuated. Let's define this as the *evacuation time*. In the *quickest flow problem*, we are given a flow value f. We want to *minimize* the time T in which a feasible flow of value at least f can be sent from sources to sinks.

4 The Single Source Single Sink Problem

In this section, we focus on the single source single sink evacuation (SSEP) problem. In real life, single source single sink evacuation problem has many applications. For example, if all the people are in an auditorium, and there is only one exit in the building, we want to evacuate people as soon as possible, when there is an emergency. Throughout the rest of this paper, s denotes the source and t denotes the sink. Before proceeding further let's have some definitions.

Definition 1. Transit time of a path *is the sum of the transit times of all the edges in P from s to t, and is denoted as $T(P)$.*

Definition 2. Destination arrival time of a path *is the time required by a person to move from s to t using path P subject to prior reservations, and is denoted as $DA(P)$. In other words, we can say that $DA(P)$ is the sum of $T(P)$ and any intermediate delay. Note that $DA(P) \geq T(P)$.*

Definition 3. Capacity of a path *is the minimum of the capacities of all nodes and edges present in the path P, and is denoted by $C(P)$.*

Definition 4. *A node (edge) on a path P is called* saturated *if the capacity of the node (edge) equals the capacity of P.*

Definition 5. *Two paths P_1 and P_2 are said to be* distinct *if $V_1 \neq V_2$ or $E_1 \neq E_2$, where V_1, V_2 are the set of vertices and E_1, E_2 are the set of edges on the paths P_1 and P_2 respectively.*

4.1 Limitation of QPER Algorithm for SSEP

Using the concept of combined evacuation time, Min et al. [11] gave an algorithm $QPER$ for the single source single sink evacuation problem. Their algorithm works well when we have already discovered k edge-disjoint paths. In $QPER$, paths from s to t are added one by one in ascending order of quickest paths, and new CET is calculated after each path addition. But after addition of a path, the new CET may be less than the transit time of a previously added path. In that case, we have to delete those paths which have higher transit time than the current CET. This in turn increases the running time, since the addition of paths is not consistent.

We overcome the above limitations of the algorithm by adding paths in increasing order of transit time in each iteration till the transit time of the currently discovered path exceeds the CET of the previously added set of paths. Note that, we need not discover all possible paths from source to sink, since unlike $QPER$, if a path is added in any iteration, it will remain till the end. The CET after each iteration will be monotonically non-increasing.

4.2 Modified Algorithm for SSEP When We Are Given k Edge-Disjoint Paths

Let $P_1, P_2, \ldots, P_k$ be k edge-disjoint paths from s to t in ascending order of their transit time, *i.e.*, $T_1 \leq T_2 \leq \ldots \leq T_k$. We define, $S_i = \{P_1, \ldots, P_i\}$. We add paths to our set of routes ($\mathcal{R}$) in the following fashion.

1. $\mathcal{R} = \{P_1\}$.
2. $CET = CET(S_1)$.
3. Start with $i = 1$ Execute step 4 and 5 till $i \leq k$ and $T_{i+1} \leq CET$.
4. Add path P_{i+1} to R.
5. $CET = CET(S_{i+1})$ and $i \leftarrow i + 1$.
6. Return $\mathcal{R}$.

Lemma 1. *If $S_j = \{P_1, P_2, \ldots, P_j\}$, $j \leq k$ is returned as $\mathcal{R}$ by the above algorithm then*

1. $T_{l+1} \leq CET(S_l), 1 \leq l < j$
2. $CET(S_1) \geq CET(S_2) \geq \ldots \geq CET(S_j)$
3. $CET(S_j) \leq CET(S_l), j < l \leq k$.

Proof. Directly follows from the algorithm.

Lemma 2. *If $S_j = \{P_1, P_2, \ldots, P_j\}$, $j \leq k$ is returned as $\mathcal{R}$ by above algorithm then $T_1 \leq T_2 \leq \ldots \leq T_j \leq CET(S_j) \leq CET(S_{j-1}) \leq \ldots \leq CET(S_1)$*

Proof. Here $T_1 \leq T_2 \leq \ldots \leq T_j$ and by Lemma 1 $CET(S_j) \leq CET(S_{j-1}) \leq \ldots \leq CET(S_1)$. So, the only thing remains to prove is $T_j \leq CET(S_j)$. Let by contrary assume that $T_j > CET(S_j)$. By putting formula for $CET(S_{j-1})$ from Eq. (1) and then solving we get $T_j > CET(S_{j-1})$. By Lemma 1, $T_j \leq CET(S_{j-1})$. This is a contradiction.

Lemma 3. *If $S_j = \{P_1, P_2, \ldots, P_j\}$, $j \leq k$ is returned as $\mathcal{R}$ by above algorithm then $CET(S_j) \leq CET(S_j \setminus \{P_i\})$, $2 \leq i \leq j$.*

Proof. We will prove this statement by contradiction. Let $CET(S_j) > CET(S_j \setminus \{P_i\})$, which implies $T_i > CET(S_j)$ by putting formula for CET from equation-1. It is not possible by Lemma 2. Hence the claim holds.

Remark 1. The addition of paths by the above algorithm is consistent, i.e. if a path is added then it will remain till the end of the algorithm execution.

4.3 An Important Observation

In Fig. 2, ordered pair (C, T) denotes capacity and transit time of an edge. There are two paths P_1 and P_2 between s and t.

$P_1 : s - B - C - E - G - t$, $C(P_1) = 4$, $T(P_1) = 19$.
$P_2 : s - A - C - E - F - t$, $C(P_2) = 6$, $T(P_2) = 23$.

P_1 and P_2 are not edge-disjoint, but common edge CE has capacity of 10 i.e. $C(P_1) + C(P_2) = C(CE)$. So, flow can be sent through P_1 and P_2 in parallel and we may think like we have two copies of edge CE one having capacity 4, dedicated for P_1 and other one having capacity 6, dedicated for P_2. We name such set of paths as "virtually edge disjoint". Now it is easy to observe that to apply the formula of combined evacuation time on a set of paths, defined in Eq. (1), the necessary condition is they should be virtually edge disjoint rather than edge disjoint.

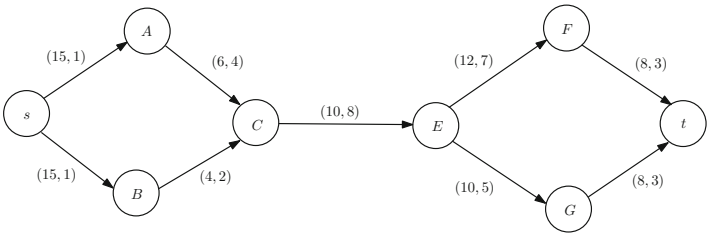

Fig. 2. An example to show that parallel flows can be sent on non edge-disjoint paths.

4.4 Our Algorithm for SSEP

The main idea of the algorithm is to find set of virtually edge disjoint paths one by one and calculate CET as in Sect. 4.2 after each path addition till it satisfies a required condition.

We discover paths one by one in the order of their transit time as follows. We find path P_1 along with its capacity C_1 having minimum transit time and decrease capacities of each node and path of P_1 by its capacity C_1 permanently and delete saturated nodes and edges. Let's say we have already added paths $\{P_1,\ P_2, \ldots,\ P_i\}$, $i \geq 1$, and updated the capacities of nodes and edges along with deletion of required saturated nodes and edges. Note that $P_1,\ P_2, \ldots,\ P_i$ are virtually edge disjoint. Hence formula of CET can be applied. In next iteration we discover a path P_{i+1} in residual graph iff t is reachable from s and $i < p$(see line number-4 in Algorithm 1). We add the discovered path P_{i+1} iff $T_{i+1} \leq CET(S_i)$(see line number-6 in Algorithm 1). As we delete saturated nodes and edges in each iteration when a path is added we discover paths in maximum of $m + n$ iterations i.e. at max $m + n$ paths and we are not going to discover more than p paths as each path can evacuate atleast one people. So, our algorithm restricts finding exponential number of possible paths from s to t . More clearly we discover at most $min(m + n, p)$ paths.

Here one may think of we are adding paths only based on transit time without considering capacity. Note that selection of a path for addition is based on transit time, addition of selected path is done if its transit time less than or equal to previously calculated CET, which is function of both capacities and transit times of previously added paths. So, our addition of paths to the solution is based on both transit time and capacities of paths implicitly.

4.5 Running Time Analysis of SSEP

From the above discussion it is clear that at most $min(m + n, p)$ paths will be discovered and equivalently our algorithm runs for at most $min(m + n, p)$ iterations. As each path discovery can be done in $O((m + n \log n)$ time, using well known Dijkstra algorithm for shortest path, our entire algorithm requires $O(\min(m + n, p)(m + n \log n)$ time. Assuming $m = O(n)$, this becomes $O(\min(n, p) \cdot n \log n)$, which is always at most $O(pn \log n)$. Recall that the time-complexity of CCRP is $O(pn \log n)$. Hence, SSEP always performs faster than CCRP. In real life, the number of evacuees is much larger than the number of vertices, so SSEP runs much faster than CCRP.

4.6 CCRP Algorithm for SSEP and Some Observations

CCRP [8] is an industry standard algorithm. Many studies have shown that the quality of solution produced by CCRP is better than most heuristic algorithms. We present the CCRP algorithm in simplified form, when there is a single source and a single sink.

Algorithm 1. Single Source Single Sink Evacuation Route Planner (SSEP)

Input: A graph $G(V, E)$ representing the network with designated source $s \in V$ and sink $t \in V$. Every node $v \in V$ has an occupancy and maximum capacity. Every edge $e \in E$ has a maximum capacity and transit time. Initially, all persons are in s.
Output: Evacuation route plan for each person.

begin
  Initialize $\mathcal{R} = \emptyset$ and $CET = \infty$.
  Initialize $i \leftarrow 0$.
  while *(t is reachable from s) and number of discovered paths* $\leq p - 1$ **do**
    Find the shortest path P_{i+1} from s to t in $G(V, E)$and let T_{i+1}, C_{i+1} be its transit time and capacity respectively.
    if $T_{i+1} \leq CET$ **then**
      $\mathcal{R} = \mathcal{R} \cup \{P_{i+1}\}$.
      $CET = CET(S_{i+1})$.
      Reduce capacity of each node and each edge of P_{i+1} by C_{i+1}.
      $V = V \setminus \{v : v$ is a saturated node of $P_{i+1}\}$.
      $E = E \setminus \{e : e$ is a saturated edge of $P_{i+1}\}$.
    end
    else
      break.
    end
  end
  Let $\mathcal{R} = \{P_1, P_2, \ldots, P_k\}$.
  Send x_i persons via $P_i, 1 \leq i \leq k$, where $T_i + \lceil \frac{x_i}{C_i} \rceil - 1 = CET$.
end

1. s is added to the priority queue. The nodes in priority queue are ordered based on the distance calculated from s during algorithm execution.
2. While there are evacuees in s, find the path P having minimum destination arrival time from s to t taking the capacity of the various nodes and edges into consideration.
3. Find capacity of P and reserve capacity along the path for a group of size equal to the minimum capacity.
4. If there are evacuees left at s, go to step 2.

Definition 6 (Group Size of a Path). *In each iteration of CCRP one path (say P_i) from s to t is discovered along with maximum number of people that can be evacuated through that path. This is defined as the* group size *of P_i for this iteration.*

For the below sections we denote T_i, C_i as transit time and group size of path P_i respectively.

Observation 1. *Let's consider execution of single source(s) single sink(t) evacuation network by CCRP algorithm. Let $P_1, P_2, \ldots, P_k$ be distinct paths(not necessarily edge-disjoint) from s to t discovered by CCRP such that $T_1 \leq T_2 \leq \ldots \leq T_k$. Here $A_i(T)$ is any permutation of $P_1(T), P_2(T), \ldots, P_i(T)$ and $P_j(T)$ is the path P_j with destination arrival time T.*

Phase 1*:* $A_1(T_1), A_1(T_1 + 1), \ldots, A_1(T_2 - 1)$
...

Phase i: $A_i(T_i), A_i(T_i+1), \ldots, A_i(T_{i+1}-1), i < k$
...
Phase k: $A_k(T_k), A_k(T_k+1), \ldots, A_k(T_k+\epsilon-2), A_k(T_k+\epsilon-1)$.

Here ϵ is the maximum number of times any path is discovered in phase k. Note that $\epsilon \geq 1$ as P_k is discovered at least once.

Number of times any path discovered in phase-k is either ϵ or $\epsilon - 1$. It is because of the following argument. By definition of ϵ there exists a path (say P_m) discovered ϵ number of times. Let P_l is a path discovered less than $\epsilon - 1$ number of times. In this case CCRP algorithm would have returned P_l instead of P_m, because using path P_l some people can reach destination before or at time $T_k + \epsilon - 2$ and P_m has earliest destination arrival time of $T_k + \epsilon - 1$.

Consider the point when all k paths have been returned $\epsilon - 1$ times in phase k. Now we may not have enough evacuees such that CCRP will return each path once. We can add some virtual evacuees such that we will use all the paths exactly ϵ times in phase-k and for simplicity we can say ϵ is the number of times path P_k is returned by CCRP.

Here it is easy to note that evacuation egress time $T_{Evac}^{CCRP} = T_k + \epsilon - 1$ and it is independent of permutation of paths in any $A_i(T)$. So, fix a permutation i.e. $A_i(T) = P_1(T), P_2(T), \ldots, P_i(T)$. Fixing up this permutation doesn't affect the solution, but it will make the analysis easier.

Observation 2. *Let $P_1, P_2, \ldots, P_k$ be distinct paths(not necessarily edge-disjoint) from s to t discovered by CCRP such that $T_1 \leq T_2 \leq \ldots \leq T_k$. Here P_i is the shortest path discovered after deletion of saturated nodes/edges of $P_1, P_2, \ldots, P_{i-1}$.*

Remark 2. Algorithm 1 finds a path even after we have deleted saturated nodes and edges of all previously discovered path, if it satisfies the conditions given on line numbers 4 and 6.

Observation 3. *Let's consider the sequence of paths as in Observation 1 with the fixed permutation of each $A_i(T)$ as explained. A path P_i may be returned in many iterations of CCRP. Group size returned in all iterations are equal possibly except last time when P_i is discovered(in phase k) in case we don't have enough evacuees left at s. This type of situation might happen only once as we are dealing with single source single destination network and it can happen in phase k after or while discovery of P_k for the first time. In such cases we can add some virtual evacuees to s so that group size of a path remains same in all iterations. It will not affect evacuation egress time but it will make the analysis easier.*

Remark 3. We can represent each path discovered by CCRP as an ordered pair of path and its group size. Algorithm 1 returns a path with maximum number of people who can travel by that path at any time. As each path is discovered only once, we can represent each path along with the capacity as an ordered pair.

4.7 Analysis of Algorithm 1

Lemma 4. *Let $(P_1, C_1), (P_2, C_2), \ldots, (P_k, C_k)$ be distinct paths (not necessarily edge-disjoint) from s to t in order of their transit time discovered by CCRP.*

1. *Number of iterations that will return path P_i is $T_k - T_i + \epsilon$, $1 \leq i \leq k$, where ϵ denotes number of iterations that returns path P_k.*
2. *Number of iterations that will return path P_i before phase j is $T_j - T_i$, where $i \leq j \leq k$.*
3. *The same paths will be returned by Algorithm 1, and $T_1 \leq T_2 \leq \ldots \leq T_k$.*

Proof. Parts (1) and (2) directly follows from Observation 1. For part (3), by induction we can prove that algorithm 1 finds each path $P_j, 1 \leq j \leq k$ with available capacity C_j.

Base Case: $j = 1$ i.e. (P_1, C_1) is added by Algorithm 1. This is obvious.

Inductive Step: Suppose paths $(P_1, C_1), \ldots, (P_j, C_j), 1 \leq j < k$ have been added by Algorithm 1. We have to prove that Algorithm 1 will also add (P_{j+1}, C_{j+1}).

Part 1: From Observation 2, P_{j+1} is the shortest path from s to t in residual graph i.e. if we delete saturated node(s) and/or edge(s) of the paths $P_1, P_2, \ldots, P_j$. Algorithm 1 also adds paths one by one after deleting saturated node(s) and/or edges(s) of previously discovered paths. So, structure of the graph remains same after addition of these j paths both in CCRP and Algorithm 1. So, P_{j+1} is also the best path w.r.t. transit time in residual graph according to Algorithm 1. As P_{j+1} is the best path in residual network either no paths will be added or P_{j+1} will be added to set of routes in Algorithm 1.

Let by contrary assume that Algorithm 1 doesn't add path P_{j+1} i.e. Algorithm 1 does not add any path. Clearly it may happen due to one of the two reasons i.e. either t is not reachable from s or number of paths discovered $= p$(line number-4 in Algorithm 1) or $T_{j+1} > CET(S_j)$(line number-6 in Algorithm 1).

Case 1(a): (t is not reachable from s)
As CCRP is able to find path P_{j+1}, t is reachable from s. Contradiction!

Case 1(b): (Number of paths discovered $= p$)
It is clear from CCRP Algorithm given in Sect. 4.6 that it does not discover more than p paths as in each path at least one people will be evacuated. As CCRP finds path P_{j+1}, number of paths discovered before discovery of P_{j+1} by Algorithm 1 can't be more than $p - 1$.

Case 2: ($T_{j+1} > CET(S_j)$)
Just come back to the point when CCRP adds path (P_{j+1}, C_{j+1}) for the first time. It can happen only in phase $j+1$. From Lemma 4 P_i is returned in $T_{j+1} - T_i, 1 \leq i \leq j < k$, iterations before phase $j+1$. As P_{j+1} discovered in phase $j+1$ for the first time total number of people evacuated through P_i before discovery of

P_{j+1} is at least $T_{j+1} - T_i$. As group size of path P_i is C_i, total number of people evacuated before discovery of P_{j+1} is at least $\sum_{i=1}^{j} C_i(T_{j+1} - T_i)$. As CCRP adds the path P_{j+1} we can say that still there are people to be evacuated. Also from Observation 3 virtual evacuees are added while or after addition of path P_k. So, total number of people evacuated before discovery of P_{j+1} is strictly less than p. Mathematically $\sum_{i=1}^{j} C_i(T_{j+1} - T_i) < p$, which implies $T_{j+1} \leq CET(S_j)$. Contradiction!

Part 2: Now one thing remains to prove is available capacity of the path P_{j+1} returned by Algorithm 1 is also C_{j+1}. If P_{j+1} doesn't share any node or edge with previously discovered path we are done. So, assume that there is some node or edge x which is common to both P_{j+1} and some P_i, $1 \leq i \leq j$. Here we argue considering x as a node and argument for x as an edge is same. Let t_n^k denotes time required to travel from s(source) to node n via path P_k with out intermediate delay. Observe that $t_x^{j+1} \geq t_x^i$. From observation 1 P_{j+1} is discovered in phase $j+1$ for the first time by CCRP algorithm. In phase $j+1$ consider $A_{j+1}(T_{j+1})$. P_i has been discovered once before discovery of P_{j+1} with its destination arrival time T_{j+1} i.e. it has made a reservation of C_i at x for the time instance t_x^{j+1} at node x. Now arrival time of evacuees via P_{j+1} to x is also t_x^{j+1}. At t_x^{j+1} we can not use that capacity of C_i for evacuees routing via P_{j+1}. In other words as if node x has dedicated capacity of C_i at time t_x^{j+1} for evacuees routing via P_i and that can't be used by evacuees routing via P_{j+1}. Here we have not assumed anything on i and x. For each such i and x, P_{j+1} can't use the capacity of C_i at time t_x^{j+1} at node x. It is equivalent to permanently decrementing the capacity of such x's by corresponding C_i, because from observation 1 whenever P_{j+1} is discovered prior to that a reservation of C_i must have been done at common node x(of P_i and P_{j+1}) by path P_i. Now come back to Algorithm 1. By induction each path P_i, $i \leq j$ is returned with capacity C_i. We find path P_{j+1} by decrementing the capacity of each path by C_i permanently. So, just before addition of P_{j+1} structure of the graph remains same w.r.t. capacity both in CCRP and Algorithm 1. From this discussion we can say that capacity of path P_{j+1} returned by Algorithm 1 is C_{j+1}.

Theorem 1. *The evacuation time of the solution given by Algorithm 1 is at most as that of the CCRP Algorithm for single source and single sink.*

Proof. Let $(P_1, C_1), (P_2, C_2), \ldots, (P_k, C_k)$ be distinct paths (not necessarily edge-disjoint) from s to t in order of their transit time (neglecting delays) discovered by CCRP. By Lemma 4, Algorithm 1 also returns the same set of paths. From Observation 1, we can say that evacuation time of CCRP is $T_{Evac}^{CCRP} = T_k + \epsilon - 1$. Evacuation time of Algorithm 1 is $CET(S_k)$. Also from Lemma 4, number of people that are evacuated through P_i is $C_i(T_k - T_i + \epsilon)$. As All people have been evacuated we can write $\sum_{i=1}^{k} C_i(T_k - T_i + \epsilon) \geq p$, which implies $T_{Evac}^{CCRP} \geq CET(S_k)$.

Theorem 2. *Upper bound on the evacuation time given by CCRP (hence by Algorithm 1) for single source single sink network is* $\lfloor \frac{p}{k} \rfloor + (n-1)\tau - 1$, *where* p *is the number of evacuees,* n *is the number of nodes in the graph,* τ *is the*

maximum transit time of any edge and k is the number of paths used by CCRP (and Algorithm 1).

Proof. From Lemma 4, number of iterations executed by CCRP is $\sum_{i=1}^{k}(T_k - T_i + \epsilon) \leq p$, as in each iteration at least one person will be evacuated. Hence, $T_{Evac}^{CCRP} \leq \lfloor \frac{p}{k} \rfloor + (n-1)\tau - 1$.

5 Randomized Behavior Model of People

The idea of combined evacuation time [11] can be extended by considering probabilistic behavior of people. Suppose in an evacuation, people do not follow the paths suggested by Algorithm 1 (or CCRP). Let's say with probability $\alpha > 0$ a person follows suggested path and with probability $1-\alpha$ he follows the shortest path (to the nearest exit). In this situation, we have to redistribute people via various paths. If we suggest x_i persons via $P_i, i \neq 1$, then the number of persons who will follow P_i and P_1 is αx_i and $(1-\alpha)x_i$ respectively (in expectation). The total number of people following P_1 and P_i are $x_1 + \sum_{i=2}^{k}(1-\alpha)x_i$ and $\alpha x_i, i \neq 1$ respectively. Expected time at which the last person will arrive at destination via P_1 is $T_1 + \frac{x_1 + \sum_{i=2}^{k}(1-\alpha)x_i}{C_1} - 1$. Expected time at which last person will arrive at destination via P_i is $T_i + \frac{\alpha x_i}{C_i} - 1, i \neq 1$
Let the expected evacuation time in this scenario be $E[T]$. Now we can write,

$$E[T] = \max\left(T_1 + \frac{(1-\alpha)n}{C_1} - 1, \max_{2 \leq i \leq k}\left(T_i + \frac{\alpha x_i}{C_i} - 1\right)\right).$$

$E[T]$ will be minimum when it satisfies the following equation,

$$\begin{aligned} E[T] &= T_1 + \frac{x_1 + \sum_{i=2}^{k}(1-\alpha)x_i}{C_1} - 1 \\ &= T_i + \frac{\alpha x_i}{C_i} - 1, 2 \leq i \leq k. \end{aligned} \tag{2}$$

where $\sum_{i=1}^{k} x_i = n$ and $x_i \geq 0, \forall i$. Solving the above equations we get,

$$E[T] = \frac{n + \sum_{i=1}^{k} C_i T_i}{\sum_{i=1}^{k} C_i} - 1 = CET(\{P_1, P_2, \ldots, P_k\}) \tag{3}$$

Expected evacuation time given by Eq. (3) doesn't depend on α. This is true and solution is feasible as long as $x_1 \geq 0$. But it is not always the case, specifically when $(1-\alpha)\sum_{i=2}^{k} x_i > C_1(T - T_1 + 1)$. So, implicitly evacuation time is dependent on α. In the following sections we give the algorithm that considers the randomized behavior of people along with analysis for expected evacuation time.

5.1 Lower Bound for Expected Evacuation Time

On expectation $x_1 + (1-\alpha)\sum_{i=2}^{k} x_i = \alpha x_1 + (1-\alpha)n$ number of people will be evacuated via path P_1. This is minimum when $x_1 = 0$ as $x_1 \geq 0$. So, lower bound for expected evacuation time is $T_1 + \frac{(1-\alpha)n}{C_1} - 1$.

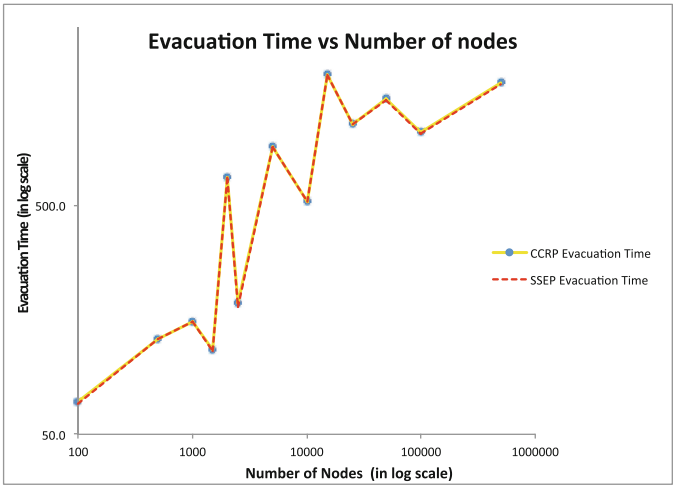

Fig. 3. Evacuation time vs number of nodes for SSEP and CCRP.

5.2 Algorithm for Randomized Behavior of People

Algorithm 2

1. Run Algorithm 1. Find CET and $x_1, x_2, \ldots, x_k$ using Eq. (2).
2. If $x_1 \geq 0$ then quit; else go to step 3. In this case, the expected evacuation time = CET.
3. Assign x'_1 to 0 and $x'_i = \frac{nx_i}{\sum_{j=2}^{k} x_j}, \forall i \neq 1$. In this case, the expected evacuation time $= T_1 + \frac{(1-\alpha)n}{C_1} - 1$.

Lemma 5. *$x'_i < x_i$, $\forall i \neq 1$, and $\sum_{i=2}^{k} x'_i = n$.*

Proof. Directly follows from the algorithm.

Lemma 6. *Above algorithm has a expected evacuation time of $CET(\{P_1, P_2, \ldots, P_k\})$ when it quits from step-2.*

Proof. In this case $x_1 \geq 0$. From the equation-4 also we can observe that $x_i \geq 0, \forall i \neq 1$. Hence the solution is feasible. So, we can safely say that the expected evacuation time is CET.

Lemma 7. *Above algorithm has a expected evacuation time of $T_1 + \frac{(1-\alpha)n}{C_1} - 1$ when it quits from step-3.*

Proof. In this case $x_1 < 0$ and by Lemma 5 $x'_i < x_i, i \neq 1$. For $i \neq 1$ x'_i number of people are suggested path P_i. Hence $T_i + \frac{px'_i}{C_i} - 1 < T_i + \frac{px_i}{C_i} - 1 < CET$, $i \neq 1$ and $T_1 + \frac{(1-\alpha)n}{C_1} - 1 > CET$.

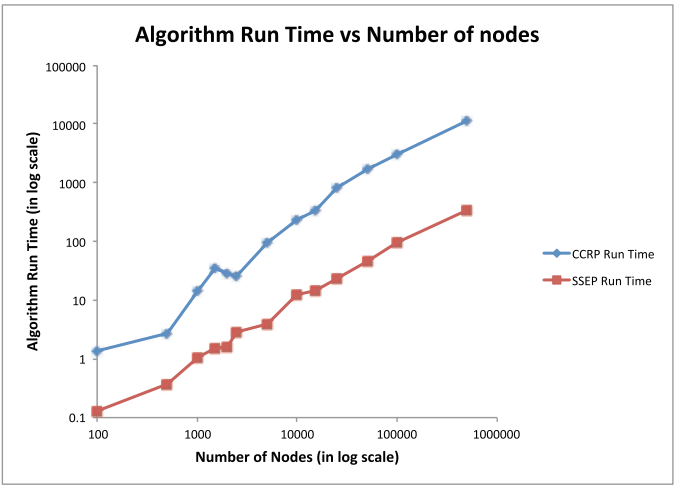

Fig. 4. Run time vs number of nodes for SSEP and CCRP.

Theorem 3. *In a single source single sink evacuation problem, if people follow the path suggested by Algorithm* 2 *with probability* α*, then the expected evacuation time is* $\max(CET, T_1 + \frac{(1-\alpha)n}{C_1} - 1)$ *and algorithm runs in* $O(\min(n, p) \cdot n \log n)$ *time.*

6 Experimental Results

6.1 Details of the Experiments

We executed the SSEP and CCRP algorithms on a Dell Precision T7600 server having an Intel Xeon E5-2687W CPU running at 3.1 GHz with 8 cores (16 logical processors) and 128 GB RAM. The operating system is Microsoft Windows 7 Professional 64-bit edition. We used the C/C++ network analysis libraries *igraph* and *LEMON* to implement the algorithms. We used *netgen* to generate synthetic graphs. The number of vertices in the graph varies from 100 to 500,000. The number of people varies from 3,000 to 120,000. The results are shown in Table 1. The graphs are plotted on a log-log scale.

6.2 Results

We show the variation of evacuation time and run time with number of nodes for SSEP and CCRP algorithms in Figs. 3 and 4 respectively. From Fig. 3, we can see that the evacuation time of SSEP is at most that of CCRP. It is evident from Fig. 4 that the running time of SSEP is much lower than that of CCRP. Hence, for all these instances SSEP clearly outperforms CCRP with respect to both evacuation time and run time. The absolute and relative amount by which SSEP performs better than CCRP is shown in Table 1.

Table 1. Comparison of evacuation time and run time of SSEP and CCRP algorithms

Number of Nodes (n)	Number of Evacuees (p)	Evacuation Time		Run Time		Improvement in SSEP over CCRP ($\frac{CCRP}{SSEP}$)	
		SSEP	CCRP	SSEP	CCRP	Evacuation Time	Run Time
100	3000	68	69	0.124	1.326	1.01	10.69
500	5000	130	130	0.358	2.73	1.00	7.63
1000	7000	155	156	1.014	14.586	1.01	14.38
1500	9000	115	117	1.466	35.443	1.02	24.18
2000	15000	661	661	1.622	29.016	1.00	17.89
2500	25000	179	186	2.761	25.739	1.04	9.32
5000	40000	903	903	3.899	93.521	1.00	23.99
10000	65000	517	520	12.012	231.535	1.01	19.28
15000	95000	1848	1853	14.025	336.946	1.00	24.02
25000	100000	1126	1128	23.134	815.682	1.00	35.26
50000	120000	1436	1446	46.69	1684.217	1.01	36.07
100000	110000	1032	1044	93.4952	3016.3005	1.01	32.26
500000	100000	1698	1720	344.341	11363.253	1.01	33.00

7 Conclusion and Future Work

In this paper, we have studied the evacuation route planning problem and given an improved algorithm for the single source single sink case. We theoretically showed that the SSEP algorithm performs better than the CCRP algorithm, both in terms of evacuation time and run time. This is also demonstrated by extensive experiments. We also analyzed a simple probabilistic behavior model of people. Here are some open problems which we would like to work in future.

- Design a system for real time monitoring of evacuation in a building using our indoor localization app [1].
- Extend this algorithm to the multiple source multiple sink case, and compare it's performance with CCRP and other algorithms.
- Develop a more sophisticated probabilistic behavior model of people for the case when they don't follow the routes suggested by the algorithm.
- Give good lower and upper bounds for the problem.

References

1. Ahmed, N., Ghose, A., Agrawal, A.K., Bhaumik, C., Chandel, V., Kumar, A.: SmartEvacTrak: a people counting and coarse-level localization solution for efficient evacuation of large buildings. In: IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pages 372–377. IEEE (2015)
2. Dressler, D., Groß, M., Kappmeier, J.-P., Kelter, T., Kulbatzki, J., Plümpe, D., Schlechter, G., Schmidt, M., Skutella, M., Temme, S.: On the use of network flow techniques for assigning evacuees to exits. Procedia Eng. **3**, 205–215 (2010)

3. Gupta, A., Sarda, N.L.: Efficient evacuation planning for large cities. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, R.R. (eds.) DEXA 2014, Part I. LNCS, vol. 8644, pp. 211–225. Springer, Heidelberg (2014)
4. Hamacher, H.W., Tjandra, S.A: Mathematical Modelling of Evacuation Problems: A State of Art. Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, Fraunhofer (ITWM) (2001)
5. Hoppe, B., Tardos, É.: Polynomial time algorithms for some evacuation problems. In: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms, pp. 433–441. Society for Industrial and Applied Mathematics (1994)
6. Kim, S., Shekhar, S., Min, M.: Contraflow transportation network reconfiguration for evacuation route planning. IEEE Trans. Knowl. Data Eng. **20**(8), 1115–1129 (2008)
7. Løvs, G.G.: Models of wayfinding in emergency evacuations. Eur. J. Oper. Res. **105**(3), 371–389 (1998)
8. Lu, Q., George, B., Shekhar, S.: Capacity constrained routing algorithms for evacuation planning: a summary of results. In: Medeiros, C.B., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 291–307. Springer, Heidelberg (2005)
9. Min, M.: Synchronized flow-based evacuation route planning. In: Wang, X., Zheng, R., Jing, T., Xing, K. (eds.) WASA 2012. LNCS, vol. 7405, pp. 411–422. Springer, Heidelberg (2012)
10. Min, M., Lee, J.: Maximum throughput flow-based contraflow evacuation routing algorithm. In: IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pages 511–516. IEEE (2013)
11. Min, M., Neupane, B.C.: An evacuation planner algorithm in flat time graphs. In: Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, p. 99. ACM (2011)
12. Skutella, M.: An introduction to network flows over time. In: Cook, W., Lovász, L., Vygen, J. (eds.) Research Trends in Combinatorial Optimization, pp. 451–482. Springer, Heidelberg (2009)
13. Song, X., Zhang, Q., Sekimoto, Y., Shibasaki, R., Yuan, N.J., Xie, X.: A simulator of human emergency mobility following disasters: knowledge transfer from big disaster data. In: AAAI Conference on Artificial Intelligence (2015)
14. Yin, D.: A scalable heuristic for evacuation planning in large road network. In: Proceedings of the Second International Workshop on Computational Transportation Science, pp. 19–24. ACM (2009)