Earliest Arrival Flow with Time Dependent Capacity Approach to the Evacuation Problems

Stevanus A. Tjandra*

*Fraunhofer Institute Techno-und Wirtschaftsmathematik (ITWM),
D-67663 Kaiserslautern, Germany.
e-mail : tjandra@itwm.fhg.de

Abstract

Evacuation problems can be modeled as flow problems in dynamic networks. A dynamic network is defined by a directed graph $G = (N, A)$ with sources, sinks and non-negative integral travel times and capacities for every arc $(i, j) \in A$. The earliest arrival flow problem is to send a maximum amount of dynamic flow reaching the sink not only for the given time horizon T, but also for any time $T^{'} < T$. This problem mimics the evacuation problem of public buildings where occupancies may not known. For the buildings where the number of occupancies is known and concentrated only in one source, the quickest flow model is used to find the minimum egress time. We propose in this paper a solution procedure for evacuation problems with a single source of the building where the occupancy number is either known or unknown. The possibility that the flow capacity may change due to the increasing of smoke density or fire obstructions can be mirrored in our model. The solution procedure looks iteratively for the shortest conditional augmenting path (SCAP) from source to sink and compute the time intervals in which flow reaches the sink via this path.

1 Introduction

In the macro approach, evacuation problems can be modeled as flow problems on the network. Since the time is a decisive parameter in evacuation

problems, dynamic network flow models are more suitable than the static ones (see e.g. [4], [5], [6], [9], [10]). A dynamic network is defined by a directed graph $G = (N, A)$ where N is the set of nodes and $A \subset N \times N$ is the set of arcs which have non-negative flow capacities and non-negative integral travel times as their attributes (see for instance [1], [2]). In the network presentation of a building, nodes may represent rooms, lobbies or intersection points. While arcs can be used to model corridors, hallways, or stairways. Some locations in the building that house a significant number of evacuees are considered to be source nodes. The building exits or safe locations that might be considered as the final destination of the evacuees' movement, are considered as sink nodes. The supply of source nodes equals to the estimation of the number of evacuees in the corresponding locations.

Dynamic network flow models have been applied to solve evacuation problems under some assumptions or objectives different from ours in this paper, e.g. constant travel time and capacity ([3], [4], [9], [10], [16]) , flow dependent exit capacities ([5], [6]), or time dependent vectors of arc costs as part of a multicriteria dynamic shortest path problem ([13], [14]). Our results relate to evacuation problems with a single source where we consider two different building environments, namely: buildings where the number of evacuees is difficult to estimate (public buildings e.g. shopping mals, theaters) and ones with known number of evacuees (e.g. office or residential buildings). In public buildings, we are interested to find the maximum number of evacuees which can be sent out within a time T. To handle this problem we work with a so-called Earliest Arrival Flow (EAF) model. EAF is the problem of maximizing flows reaching the sink not only for the alloted time T, but also for any smaller time horizon (see e.g. [7], [10], [11], [15]).

The main contribution of this paper is to incorporate the changing of arc capacities over time, e.g. due to smoke or fire, into this EAF problem. We assume that evacuees may wait only in the source node. The solution procedure is applied to solve the evacuation problem with known occupancy number and with objective to minimize the evacuation time with respect to the egress time. The latter problem is known as the quickest flow problem (see [3], [7], [11]).

In the next section we will formally introduce the continuous time dynamic network formulation of EAF (CTEAF). In Section 3 and 4, we will introduce the idea of conditional augmenting path and detail procedure to find this path. Since the solution of EAF also solves maximum dynamic network flow problem which has dual relation with dynamic cut, Section 5 will discuss about continuous time dynamic cut and detail procedure to obtain the minimum dynamic cut from the CTEAF's solution. To show how the procedure work, in Section 6 we present the illustrative example in detail.

The last section, Section 7, contains some interrelations between CTEAF and the quickest flow problem which is used to find the minimum egress time of evacuation problem with known occupancies.

2 Problem Definition

Given the network $G = (N, A)$ where node s and d is the source and the sink node, respectively. We denote by n and m the cardinality of N and A, respectively. With each arc $(i, j) \in A$ we associate a nonnegative integer number λ_{ij} which gives the time to traverse the arc, and a nonnegative bounded measurable function b_{ij} on $[0, T]$ with finite breakpoints which gives its flow capacity. T, the time horizon of interest is determined a priori. It is assumed that no node except the source has storage capacity, i.e. no flow may wait in any node but the source node. The value $x_{ij}(t)$ defines the nonnegative rate of flow leaving node i at time t, consequently arriving at node j at time $t + \lambda_{ij}$. The continuous-time EAF problem is thus formulated as follows.

$$\text{(CTEAF)} \qquad \max V_{T'}(x) = \int_0^{T'} \sum_{i \in N} [x_{id}(\tau - \lambda_{id}) - x_{di}(\tau)] d\tau, \quad T' \in [0, T] \tag{1}$$

$$\sum_{(j,i) \in A} x_{ji}(t - \lambda_{ji}) - \sum_{(i,j) \in A} x_{ij}(t) = 0, \ t \in (0, T], i \in N - \{s, d\}, \tag{2}$$

$$0 \leq x_{ij}(t) \leq b_{ij}(t), \ \forall (i, j) \in A, \ t \in [0, T]. \tag{3}$$

This problem is solvable since the feasible set is compact and nonempty in $L_\infty^m[0, T]$.

3 Conditional Augmenting Path

A solution procedure for this problem is a dynamic version of the well-known shortest augmenting path algorithm of maximum static network flows (see, for instance [1]). In the classical maximum flow problem we look for a shortest path from s to d ($s - d$ path) in the residual network. Such a path is known as the shortest augmenting path. In the case of the dynamic version we have to consider the availability of the shortest augmenting path over time.

Instead of working with dynamic network directly for finding the shortest augmenting path, we keep working with static network but with additional attribute S_{ij} (besides capacity and travel time) on each arc $(i, j) \in N \times N$.

The admissible time S_{ij} for arc $(i,j) \in N \times N$ is defined as the set of time intervals when the arc can cary residual by increasing or decreasing the current flow as follows.

Definition 3.1

$$S_{ij} := \begin{cases} \{t : b_{ij}(t) - x_{ij}(t) > 0\} & ,\text{if } (i,j) \in A \\ \{t + \lambda_{ji} : x_{ji}(t) > 0\} & ,\text{if } (j,i) \in A \end{cases} \tag{4}$$

The first alternative represents the set of times when it is possible to increase flow from i to j, while the second one represents the set of times when it is possible to decrease flow from j to i by sending back some flows from i to j.

Since the flow augmentation on arc (i,j) is possible only when $S_{ij} \neq \emptyset$, the residual network is conditioned by the set S_{ij}. Therefore we define the residual network as follows.

Definition 3.2 *Residual network with respect to the feasible dynamic flow x is defined as $G_x := (N, A_x^+ \cup A_x^-)$ with arc set $A_x^+ := \{(i,j) : (i,j) \in A,\ S_{ij} \neq \emptyset\}$ and $A_x^- := \{(i,j) : (j,i) \in A,\ S_{ij} \neq \emptyset\}$. The residual travel times are*

$$\lambda_{ij}^x := \begin{cases} \lambda_{ij} & ,(i,j) \in A_x^+ \\ -\lambda_{ji} & ,(i,j) \in A_x^- \end{cases} \tag{5}$$

Figure 1 shows the residual network with residual travel times.

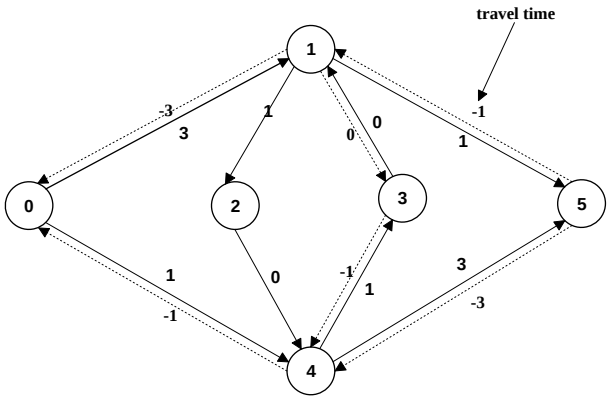

Figure 1: Residual Network G_x

Since the arc capacity is not constant over time, we can send additional flow from the source to the sink along path P only when we sure that this flow can reach the sink. This condition means that the residual capacity

of each arc $(i,j) \in P$ at the time when the flow reaches this arc must be positive. Suppose $P = \{s = v_1, v_2, \ldots, v_k = d\}$ is a $s-d$ path in the residual network G_x. Let $\pi_{v_l}^{v_{l+1}}(P)$ be the label measuring the distance of v_l to d along P with respect to travel time $\lambda_{v_l v_{l+1}}$. Then

$$S_{v_l}^{v_{l+1}}(P) := \{t + \pi_{v_l}^{v_{l+1}}(P) : t \in S_{v_l,v_{l+1}}\} \cap [0,T]$$

is the set of arival times of flows at d within the time T that leave v_l at time $t \in S_{v_l v_{l+1}}$ and move along P, under the assumption that increasing the flow is always possible. Increasing flow along path P is possible only when the arrival time at the next nodes $v_{l+q}, q = 1, \ldots, k-l-1$ of flows which leave v_l at time $t \in S_{v_l,v_{l+1}}$, $\forall l = 1, \ldots, k-2$ is also in $S_{v_{l+q},v_{l+q+1}}$. We define recursively for $l = 1, \ldots, k-1$

$$S_{v_l}^{*v_{l+1}}(P) := S_{v_l}^{v_{l+1}}(P) \cap S_{v_{l+1}}^{*v_{l+2}}(P)$$

with $S_{v_k}^{*v_{k+1}}(P) = [0,T]$. $S_{v_l}^{*v_{l+1}}(P)$ represents the set of arrival times at the sink node when flows from node v_l reach d with considering the availability of the next arcs along P in the residual network. Figure 2 illustrates the transfer from $S_{v_l v_{l+1}}$ to $S_{v_l}^{*v_{l+1}}(P)$.

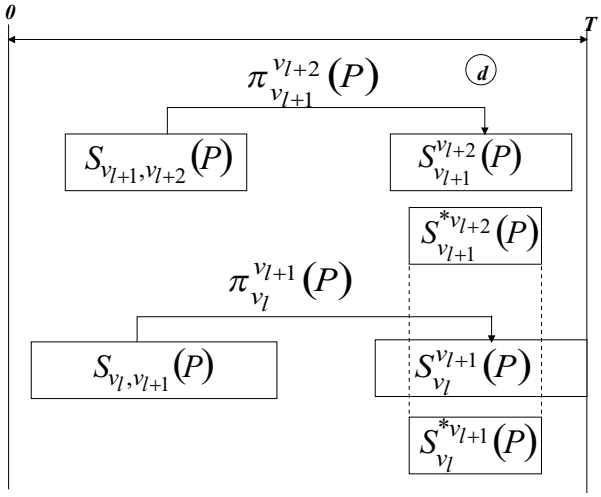

Figure 2: Determination of $S_{v_l}^{*v_{l+1}}(P)$ from $S_{v_l v_{l+1}}(P)$

Definition 3.3 *Node j is said to be i-reachable if there is a finite path of nodes and arcs connecting node i to node j through which a feasible increase in flow (from i to j) can be made.*

We say that the sink is v_l-reachable via path P in G_x when $S_{v_l}^{*v_{l+1}}(P) \neq \emptyset$. Hence, increasing flow from the source to the sink along path P is possible only when $S_{v_1}^{*v_2}(P) \neq \emptyset$. This condition defines the so-called *conditional flow augmenting path.*

Definition 3.4 *Path P is a conditional (flow) augmenting path (CAP) in G_x if*

$$S_P := S_{v_1}^{*v_2}(P) \neq \emptyset$$

We define the set of forward arcs and the set of backward arcs of P as $P^+ := P \cap A_x^+$ and $P^- := P \cap A_x^-$, respectively.

Definition 3.5 *The set of departure times S_{ij}^P of flows on arc (i,j) sent along P for which the arrival time at d is in S_P, is defined as*

$$S_{ij}^P := \{t - \pi_i^j(P) \ : \ t \in S_P\} \tag{6}$$

The residual capacity of each arc of P thus can be formulated as follows.

$$b_{ij}^x(t) := \begin{cases} b_{ij}(t) - x_{ij}(t) & ,\text{if } (i,j) \in P^+ \\ x_{ji}(t - \lambda_{ji}) & ,\text{if } (i,j) \in P^- \end{cases} \ , \ t \in S_{ij}^P \tag{7}$$

These residual capacities must be translated forward from their own domain S_{ij}^P to the common domain S_P in order to find the minimum residual capacity which defines the maximum increment along P. The forward translation $b_{ij}^{'x}$ of b_{ij}^x for all $(i,j) \in P$ can be obtained as follows.

$$b_{ij}^{'x}(t) := b_{ij}^x(t - \pi_i^j(P)) \ , t \in S_P, \ \forall (i,j) \in P \tag{8}$$

Next we compute the maximum increment functions ϵ_P^+, ϵ_P^-, and ϵ_P on S_P as the following.

$$\begin{aligned} \epsilon_P^+(t) &:= \min\{b_{ij}^{'x}(t) \ : \ (i,j) \in P^+\}, \ t \in S_P \\ \epsilon_P^-(t) &:= \min\{b_{ij}^{'x}(t) \ : \ (i,j) \in P^-\}, \ t \in S_P \\ \epsilon_P(t) &:= \min\{\epsilon_P^+(t), \epsilon_P^-(t)\}, \ t \in S_P \end{aligned} \tag{9}$$

The increment functions are translated back to S_{ij}^P in order to accomodate the increment on each $(i,j) \in P$

$$\epsilon_{ij}(t) := \begin{cases} \epsilon_P(t + \pi_i^j(P)) & ,\text{if } \ t \in S_{ij}^P \\ 0 & ,\text{otherwise} \end{cases} \ , \ t \in [0,T] \tag{10}$$

The new flows for each time $t \in [0,T]$ thus can be obtained as follows.

$$x_{ij}(t) := \begin{cases} x_{ij}(t) + \epsilon_{ij}(t) & ,\text{if } (i,j) \in P^+ \cup \{(d,s)\} \\ x_{ij}(t) - \epsilon_{ji}(t + \lambda_{ij}) & ,\text{if } \ (j,i) \in P^- \\ x_{ij}(t) & ,\text{otherwise} \end{cases} \ , \ t \in [0,T] \tag{11}$$

4 Finding the Shortest Conditional Augmenting Path

In order to find the available path from any node $i \in N$ to d, we work with the network $G_x^R := (N, A_x^R)$. In G_x^R, the arc set A_x^R is obtained by reversing the direction of all arcs in A_x. By working with reverse network, it means that we start moving from the sink to the source for finding the SCAP. We keep arc connecting node i to node j as long as $S_{ij} \neq \emptyset$. The residual network is not purely static since the arc $(i,j) \in A_x$ is passable only at time $t \in S_{ij} \subset [0,T]$.

During the process of finding the shortest conditional augmenting path, we may encounter the shortest path that reach node $j \neq s$ at time $t' \notin S_{jk},\ \forall (j,k) \in A_x^R$, i.e. this shortest path can reach node j but can not reach s. Therefore, we can not update the large distance with the smaller one when finding the shortest conditional augmenting path, as in the label correcting algorithm for the static network. The following example shows this phenomenon.

Example 4.1
Consider the reverse network of G_x shown in Figure 1. The admissible time for each arc is defined as follows.
$S_{40} =]5,8]\ ;\ S_{51} =]0,2]\ ;\ S_{04} =]1,6]\ ;\ S_{43} =]2,3] \cup]6,7]\ ;\ S_{31} =]2,3] \cup]6,7]\ ;$
$S_{15} =]3,8]\ ;\ S_{01} =]3,6]\ ;\ S_{45} =]5,8]$; others are $[0,8]$.
In the beginning we label the distance of node 4 with 3 units that corresponds to the path $P = (4,5)$. Later on we find that there is a shorter path from node 4 to the sink, that is the path $P^{'} = (4,3,1,5)$ with distance equal to 2 units. Now, Consider any $0-5$ path $P^{''}$ in G_x^R. Since $S_0^i(P^{''}) \cap S_1^{*5}(P^{'}) = S_0^i(P^{''}) \cap]1,3] = \emptyset,\ \forall (i,0) \in A_x^R$, there is no CAP from source 0 to sink 5 contains path $P^{'}$. On the other hand, if we keep the longer distance of node 4, i.e. keep the path P instead of $P^{'}$, then we can continue this path to reach the source, that is via path $(0,1,2,4,5)$ with total distance 7 units. Hence, updating P with $P^{'}$ will give the wrong result.

In the case of time dependent capacities, we update the large distance of node j from node i via path P, $\pi_j^i(P)$, with the new distance from node i° via path P° only when $\pi_j^i(P) \geq \pi_j^{i^\circ}(P^\circ)$ and $S_j^{*i}(P) \subset S_j^{*i^\circ}(P^\circ)$ as explained by Lemma 4.1.

Lemma 4.1 *Suppose there are two paths from j to d, P_{jd} and P_{jd}° with $(j,i) \in P_{jd}$ and arc $(j,i^\circ) \in P_{jd}^\circ$. Let $\pi_j^i(P_{jd})$ and $\pi_j^{i^\circ}(P_{jd}^\circ)$ be the distance of j to d on path P_{jd} and P_{jd}°, respectively, with $S_j^{*i}(P_{jd})$ and $S_j^{*i^\circ}(P_{jd}^\circ)$ the corresponding arrival times at d. Moreover, assumes that $\pi_j^i(P_{jd}) \geq \pi_j^{i^\circ}(P_{jd}^\circ)$*

and $S_j^{*i}(P_{jd}) \subset S_j^{*i^\circ}(P_{jd}^\circ)$. *If there is a* $s-d$ *conditional augmenting path* $P := P_{sj} \cup P_{jd}$, *then there is also a path* $P^\circ = P_{sj} \cup P_{jd}^\circ$ *with shorter time distance than that of path* P.

Proof:
Since P is a $s-d$ CAP, $S_P \neq \emptyset$. By definition, $S_P \subset S_l^j(P_{sj}) \cap S_j^{*i}(P_{jd})$ for any $l \in N$ with $(l,j) \in P_{sj}$. Since $S_j^{*i}(P_{jd}) \subset S_j^{*i^\circ}(P_{jd}^\circ)$, we obtain $S_P \subset S_l^j(P_{sj}) \cap S_j^{*i^\circ}(P_{jd}^\circ)$. Therefore, we may compose an $s-d$ path P° with $S_{P^\circ} \supset S_P$ from path P as described in the lemma. Since $\pi_j^i(P_{jd}) > \pi_j^{i^\circ}(P_{jd}^\circ)$ then the time distance of P° is shorter than that of P. ■

The SCAP procedure maintains a set of distance labels π and the time set S^* which are updated iteratively. The distance label π_j^i is either ∞ and $S_j^{*i} = \emptyset$, indicating that we have yet to discover an augmenting path from the node j to the sink node, or it is equal to the length of some augmenting path from the node j to the sink with $S_j^{*i} \neq \emptyset$. For each node j, we maintain a list of predecessor indices Z_j which records some node prior to node j via some augmenting paths. We write the q-th predecessor of node j via the q-th $j-d$ conditional augmenting path P with $(j,i) \in P$, during iteration p as $Z_j^p(q) = (i, q')$ where q' is the predecessor index of node i in the previous iteration. Moreover we denote by $S_j^{*p,q}$ and $\pi_j^{p,q}$ the set $S_j^{*i}(P)$ and $\pi_j^i(P)$, respectively. At termination, the predecessor indices allow us to trace the SCAP.

Corollary 4.1 *Consider iteration* $p+1$ *of the algorithm and any* $(i,j) \in A_x^R$. *Suppose that* $\exists\ q' \in \{1, \ldots, \mid Z_i^p \mid\}$ *with* $B := \{t + \pi_i^{p,q'} + \lambda_{ij}^x \ : \ t \in S_{ij}\} \cap [0,T] \cap S_i^{*p,q'} \neq \emptyset$. *If* $\exists\ q \in \{1, \ldots, \mid Z_j^p \mid\}$ *with* $\pi_j^{p,q} \geq \pi_i^{p,q'} + \lambda_{ij}^x$ *and* $S_j^{*p,q} \subset B$ *then we get a shorter CAP by updating the value of* $\pi_j^{p,q}$ *with* $\pi_i^{p,q'} + \lambda_{ij}^x$, *i.e.* $\pi_j^{p+1,q} = \pi_i^{p,q'} + \lambda_{ij}^x$.

Proposition 4.1 *If the capacity function is piecewise constant and changes only in the integer time units, then for any iteration* p *of the SCAP algorithm,* $\mid Z_j^p \mid \leq 3T,\ \forall j \in N$.

Proof :
From Lemma 4.1, for any $j \in N$ and for any iteration p, $\mid Z_j^{p+1} \mid = \mid Z_j^p \mid$ only when there exists $q \leq \mid Z_j^p \mid$ such that $\pi_j^{p,q} > \pi_i^{p,q'} + \lambda_{ij}^x$ and $S_j^{*p,q} \subset B$ or $\pi_j^{p,q} \leq \pi_i^{p,q'} + \lambda_{ij}^x$ and $S_j^{*p,q} \supset B$, with B as the defined in the Corollary 4.1. The first condition updates the q-th element of Z_j^p but does not increase its cardinality. The latter does not give any effect to Z_j^p. Therefore there are 3 possibilities for increasing $\mid Z_j^p \mid$.

(a) $\pi_j^{p,q} < \pi_i^{p,q'} + \lambda_{ij}^x$ and $S_j^{*p,q} \not\supset B,\ \forall q \leq \mid Z_j^p \mid$. Since the travel times are integer and not larger than T, the distance value π is bounded above by T and bounded below by $-T$. Therefore, there are at most $2T$ possibilities of $\pi_i^{p,q'} + \lambda_{ij}^x$ to be greater than $\pi_j^{p,q}$. On the other hand, suppose $S_j^{*p,q}$ and B consist of single interval $[a_1, a_2]$ and $[b_1, b_2]$, respectively. $S_j^{*p,q} \supset B$ occurs when $a_1 \leq b_1$ and $a_2 \geq b_2$, i.e., $S_j^{*p,q} \not\supset B$ occurs when $a_1 > b_1$ or $a_2 < b_2$. Since $S_j^{*p,q}$ and B must be contained in $[0, T]$ and moreover, the boundary values of $S_j^{*p,q}$ and B are also integers, there are maximum T possibilities values of b_1 and b_2 such that $a_1 > b_1$ or $a_2 < b_2$. Therefore under condition $\pi_j^{p,q} < \pi_i^{p,q'} + \lambda_{ij}^x$ and $S_j^{*p,q} \not\supset B,\ \forall q \leq \mid Z_j^p \mid$, $\mid Z_j^p \mid$ must be not larger than T.

(b) $\pi_j^{p,q} > \pi_i^{p,q'} + \lambda_{ij}^x$ and $S_j^{*p,q} \not\subset B,\ \forall q \leq \mid Z_j^p \mid$. Using similar proof of (a), $\mid Z_j^p \mid$ must be not larger than T due to this condition.

(c) $S_j^{*p,q} \not\subset B$ and $S_j^{*p,q} \not\supset B,\ \forall q \leq \mid Z_j^p \mid$. In this case, there are 2 possible structures, $S_j^{*p,q} \cap B = \emptyset$ or $S_j^{*p,q} \cap B \neq \emptyset$. Suppose $S_j^{*p,q}$ and B consist of a single interval $[a_1, a_2]$ and $[b_1, b_2]$, respectively. $S_j^{*p,q} \cap B = \emptyset$ occurs when $a_2 < b_1$ or $a_1 > b_2$. Since $S_j^{*p,q}$ and B must be contained in $[0, T]$ and the boundary values of $S_j^{*p,q}$ and B are also integers, there are maximum T possibilities values of b_1 and b_2 such that $a_2 < b_1$ or $a_1 > b_2$. The latter condition is possible only when $a_1 < b_1 < a_2 < b_2$ which gives maximum T possibilities values of b_1 and b_2.

Since those 3 possibilities are disjoint, $\mid Z_j^p \mid \leq 3T$ ■

The iterative processes of SCAP procedure stops when they satisfy the following optimality condition.

Theorem 4.1 (SCAP Optimality Condition) *Suppose P_{jd} is the conditional augmenting path from any node j to the sink d. The distance label $\pi_j^l(P_{jd}),\ (j,l) \in P_{jd}$ represents the SCAP distance if and only if for any conditional augmenting path P_{id} with $(i,j) \in A_x^R$, they satisfy the SCAP conditions*

$$\pi_j^l(P_{jd}) \leq \pi_i^k(P_{id}) + \lambda_{ij}^x,\ (i,k) \in P_{id}$$

or

$$S_j^{*i}(P_{jd}^{'}) = \emptyset,\ \forall P_{jd}^{'} \text{ with } (j,i) \in P_{jd}^{'}$$

Proof :
"⇒" : Suppose $\exists\ (i,j) \in A_x^R$ with $\pi_j^l(P_{jd}) > \pi_i^k(P_{id}) + \lambda_{ij}^x$. If $S_j^{*i}(P_{jd}^{'}) \neq \emptyset$ then it contradicts the asumption of $\pi_j^l(P_{jd})$.
"⇐" : Follows obviously from the definition of SCAP. ■

Lemma 4.2 *If the capacity function is piecewise constant and changes only in the integer time units, then the SCAP algorithm has the worst-case computational complexity* $O(nT)$.

Proof :
Due to Lemma 4.1, we update $\pi_j^{p,q}$ for any p and any $q \leq \mid Z_j^p \mid$ only when $\pi_j^{p,q} \geq \pi_i^{p,q'} + \lambda_{ij}^x$ and $S_j^{*p,q} \subset B$. The value of $\pi_j^{p,q}$ is bounded from above and from below by T and $-T$, respectively. Since each update of $\pi_j^{p,q}$ decreases it by at least 1 unit, the algorithm updates any label $\pi_j^{p,q}$ at most $2T$ times without considering the requirement $S_j^{*p,q} \subset B$. Suppose $S_j^{*p,q}$ and B consist of a single interval $[a_1, a_2]$ and $[b_1, b_2]$, respectively. Condition $S_j^{*p,q} \subset B$ ocurs when $a_1 > b_1$ and $a_2 < b_2$ which have at most T possibilities, i.e. condition $S_j^{*p,q} \subset B$ is fulfilled at most T times. Hence, we update $\pi_j^{p,q}$ under condition $S_j^{*p,q} \subset B$ by at most T times. Since in each iteration we may either increase the cardinality of Z_j^p or update the q-element of Z_j^p, with n the total number of nodes, the SCAP algorithm does at most $4nT$ assignments, i.e., the algorithm converges in $O(nT)$ time in the worst case. ■

We see that the solution procedure consists of two main parts as shown in Figure 3. The first part is to find the shortest conditional augmenting path

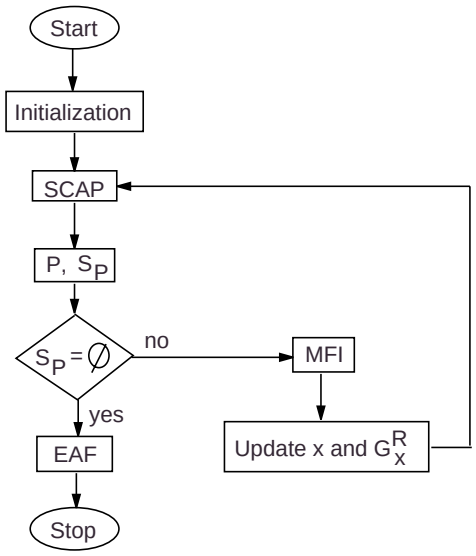

Figure 3: EAF Algorithm

P and its availability time S_P. The second one is to find the maximum flow increment (MFI) along P and to update the current flows. These two parts are repeated until no conditional augmenting path is available. At this point the current flow solves the EAF due to the following lemmas.

Lemma 4.3 *If $S_P \neq \emptyset$ then there is feasible flow with value strictly greater than that of the current flow.*

Proof :
Suppose $\{x_{ij} \mid (i,j) \in A\}$ is the current feasible flow and there is a $s-d$ path P with $S_P \neq \emptyset$. Since $S_P := S^{*v_2}_{s=v_1}(P)$ and $S^{*v_2}_{s}(P) \subset S^{*v_{l+1}}_{v_l}(P)$, $\forall (v_l, v_{l+1}) \in P$, we have $S^{*,v_{l+1}}_{v_l}(P) \neq \emptyset$. This condition implies $S_{v_l,v_{l+1}}(P) \neq \emptyset$ and moreover $\epsilon_P(t) > 0,\ t \in S_P$. Hence x_{ds} is not optimal. ■

Lemma 4.4 *If $S_P = \emptyset$ then the current flow solves EAF.*

Proof :
Suppose $\mathbf{x}$ does not solve maximum dynamic flow for some $T' < T$. Then there is a $s-d$ path P with $S_P \subset [0, T']$ which is not empty, contradicting the assumption of S_P. ■

SCAP algorithm generates conditional augmenting path with the shortest distance but it may not reach the sink at the earliest time, because there may be exist shortest CAP with active arcs latter than other arcs (i.e. the lower boundary of the corresponding admissible time is greater than the other arcs). Therefore, the result may violate FIFO (First-In First-Out) rule. EAF is obtained only when the algorithm satisfying condition in lemma 4.4.

5 Finding the Minimum Cut

Similar to the discrete time dynamic cut, the continuous time dynamic cut is defined as follows.

Definition 5.1 ([17]) *Let us denote $a_i(t)$ the continuous function of the holdover capacity at node i at time t. A continuous time $s-d$ dynamic cut is a set-valued function*

$$C_T : [0, T] \to 2^N$$

that satisfies for every $t \in [0, T]$, the source $s \in C_T(t)$ and the sink $d \notin C_T(t)$ with property that

$$A_i = \{t \ :\ i \in C_T(t)\} \cap \{t \ :\ a_i(t) > 0\} \cap\ ]0, T] \text{ is open },\ \forall i \in N - \{s, d\}$$

The definition that $a_i(t) = 0,\ \forall i \in N - \{s, d\},\ t \in [0, T]$ in our model does not violate the definition of the cut since the empty set is an open set.

At any instant of time t, any node i will either be in the source side of the cut, i.e. it is a member of $C_T(t)$, or in the sink side of the cut, i.e. it is

a member of $\overline{C}_T(t)$. We have Γ_i as the set of times $t \in [0,T]$ when i is in the source side of the cut, i.e.

$$\Gamma_i := \{t \ : \ i \in C_T(t), \ t \in [0,T]\}$$

and we have Γ_{ij} as the set of times $t \in [0,T]$ when i is in the source side of the cut and j is in the sink side of the cut, i.e.

$$\Gamma_{ij} := \{t : t \in [0, T-\lambda_{ij}], i \in C_T(t), j \notin C_T(t+\lambda_{ij})\} \tag{12}$$

As in the classical cut, the value $b_T(C_T)$ of the cut C_T is determined by the capacities of arcs which cross the cut, namely

$$b_T(C_T) = \sum_{(i,j)\in A} \int_{\Gamma_{ij}} b_{ij}(t)dt \tag{13}$$

with assumption that $a_i(t) = 0, \ \forall i \in N - \{s,d\}$ and $t \in [0,T]$. The following lemma describes the interrelation between the value of maximum dynamic flow and the value of dynamic cut.

Lemma 5.1 *Let us denote by* $\mathbf{C}_T$ *the set of dynamic cut* C_T*. For any feasible flow* x *and any dynamic cut* $C_T \in \mathbf{C}_T$*,*

$$V_T(x) \leq b_T(C_T)$$

Proof:

$$V_T(x) = \int_0^T \sum_{j\in N} \big[x_{jd}(t-\lambda_{jd}) - x_{dj}(t)\big]dt$$

Since $\sum_{(j,i)\in A} x_{ji}(t-\lambda_{ji}) - \sum_{(i,j)\in A} x_{ij}(t) = 0, \ \forall i \in N - \{s,d\}$ by constraint (2), we can modify $V_T(x)$ to be

$$\begin{aligned} V_T(x) &= \int_0^T \sum_{j\in N} \big[x_{jd}(t-\lambda_{jd}) - x_{dj}(t)\big]dt + \int_0^T \sum_{i\in N-\{s,d\}} \sum_{j\in N} \big[x_{ji}(t-\lambda_{ji}) - x_{ij}(t)\big]dt \\ &= \int_0^T \sum_{i\in N} \sum_{j\in N} \big[x_{ji}(t-\lambda_{ji}) - x_{ij}(t)\big]dt - \int_0^T \sum_{j\in N} \big[x_{js}(t-\lambda_{js}) - x_{sj}(t)\big]dt \end{aligned}$$

Since $\Gamma_s = [0,T]$, we can write

$$\begin{aligned} V_T(x) &= \int_0^T \sum_{i\in N} \sum_{j\in N} \big[x_{ji}(t-\lambda_{ji}) - x_{ij}(t)\big]dt - \\ &\quad \int_{\Gamma_s} \sum_{j\in N} \big[x_{js}(t-\lambda_{js}) - x_{sj}(t)\big]dt \end{aligned} \tag{14}$$

The transformation $t^{'} = t - \lambda_{ji}$ is applied to the first term on the right-hand side of Equation (14) followed by disregarding the negative domain of integration to yield

$$\begin{aligned}\int_0^T x_{ji}(t-\lambda_{ji})dt &= \int_0^{T-\lambda_{ji}} x_{ji}(t^{'})dt^{'} \\ &= \int_0^T x_{ji}(t^{'})dt^{'} - \int_{T-\lambda_{ji}}^T x_{ji}(t^{'})dt^{'}\end{aligned}$$

and implies

$$\begin{aligned}\int_0^T \sum_{i\in N}\sum_{j\in N}\big[x_{ji}(t-\lambda_{ji}) - x_{ij}(t)\big]dt &= \sum_{i\in N}\sum_{j\in N}\Big[\int_0^T x_{ji}(t^{'})dt^{'} - \int_{T-\lambda_{ji}}^T x_{ji}(t^{'})dt^{'} - \\ &\qquad \int_0^T x_{ij}(t)dt\Big] \\ &= -\sum_{i\in N}\sum_{j\in N}\int_{T-\lambda_{ji}}^T x_{ji}(t^{'})dt^{'}\end{aligned}$$

We obtain

$$V_T(x) = \int_{\Gamma_s}\sum_{j\in N}\big[x_{sj}(t) - x_{js}(t-\lambda_{js})\big]dt - \sum_{i\in N}\sum_{j\in N}\int_{T-\lambda_{ji}}^T x_{ji}(t)dt$$

Since $\Gamma_d = \emptyset$ by the definition of the cut, we have

$$\int_{\Gamma_d}\sum_{j\in N}\big[x_{dj}(t) - x_{jd}(t-\lambda_{jd})\big]dt = 0 \tag{15}$$

and by constraint (2), we have

$$\sum_{i\in N-\{s,d\}}\int_{\Gamma_i}\sum_{j\in N}\big[x_{ij}(t) - x_{ji}(t-\lambda_{ji})\big]dt = 0 \tag{16}$$

By adding (15) and (16) to $V_T(x)$, we obtain

$$V_T(x) = \sum_{i\in N}\int_{\Gamma_i}\sum_{j\in N}\big[x_{ij}(t) - x_{ji}(t-\lambda_{ji})\big]dt - \sum_{i\in N}\sum_{j\in N}\int_{T-\lambda_{ij}}^T x_{ij}(t)dt \tag{17}$$

Transform $t^{'} = t - \lambda_{ji}$ of the second term on the right-hand side of Equation (17) to yield

$$\int_{\Gamma_i} x_{ji}(t-\lambda_{ji})dt = \int_{\Gamma_i^{'}} x_{ji}(t^{'})dt^{'}$$

with $\Gamma_i^{'} = \{t^{'} \ : \ t^{'} \in [0, T - \lambda_{ji}],\ i \in C_T(t^{'} + \lambda_{ji})\}$
Define

$$Q1_{ij} = \{t \ : \ t \in [0, T - \lambda_{ij}],\ i \in C_T(t),\ j \in C_T(t + \lambda_{ij})\}$$
$$Q2_{ij} = \{t \ : \ t \in [0, T - \lambda_{ji}],\ i \in C_T(t + \lambda_{ji}),\ j \in \overline{C}_T(t)\}$$
$$Q3_{ij} = \{t \ : \ t \in [0, T - \lambda_{ji}],\ i \in C_T(t + \lambda_{ji}),\ j \in C_T(t)\}$$

Using these definitions, we obtain

$$\begin{aligned}\sum_{i\in N}\int_{\Gamma_i}\sum_{j\in N}\big[x_{ij}(t)\big]d(t) &= \sum_{i\in N}\sum_{j\in N}\int_{\Gamma_{ij}} x_{ij}(t)d(t) + \sum_{i\in N}\sum_{j\in N}\int_{Q1_{ij}} x_{ij}(t)d(t) + \\ &\quad \sum_{i\in N}\sum_{j\in N}\int_{\Gamma_i\cap[T-\lambda_{ij},T]} x_{ij}(t)d(t)\end{aligned}$$

and

$$\begin{aligned}\sum_{i\in N}\int_{\Gamma_i}\sum_{j\in N}\big[x_{ji}(t-\lambda_{ji})\big]d(t) &= \sum_{i\in N}\int_{\Gamma_i^{'}}\sum_{j\in N}\big[x_{ji}(t^{'})\big]d(t^{'}) \\ &= \sum_{i\in N}\sum_{j\in N}\int_{Q2_{ij}} x_{ji}(t)d(t) + \sum_{i\in N}\sum_{j\in N}\int_{Q3_{ij}} x_{ji}(t)d(t)\end{aligned}$$

Since

$$\sum_{i\in N}\sum_{j\in N}\int_{Q1_{ij}} x_{ij}(t)d(t) = \sum_{i\in N}\sum_{j\in N}\int_{Q3_{ij}} x_{ji}(t)d(t)$$

Equation (17) is reduced to

$$\begin{aligned}V_T(x) &= \sum_{i\in N}\sum_{j\in N}\Big[\int_{\Gamma_{ij}} x_{ij}(t)d(t) - \int_{Q2_{ij}} x_{ji}(t)d(t) + \\ &\quad \int_{\Gamma_i\cap[T-\lambda_{ij},T]} x_{ij}(t)d(t) - \int_{T-\lambda_{ij}}^{T} x_{ij}(t)dt\Big] \\ &= \sum_{i\in N}\sum_{j\in N}\Big[\int_{\Gamma_{ij}} x_{ij}(t)d(t) - \int_{Q2_{ij}} x_{ji}(t)d(t) - \int_{[T-\lambda_{ij},T]-\Gamma_i} x_{ij}(t)dt\Big]\end{aligned}$$

Since $0 \le x \le b$, we obtain

$$\begin{aligned}V_T(x) &\le \sum_{i\in N}\sum_{j\in N}\int_{\Gamma_{ij}} x_{ij}(t)d(t) \\ &\le \sum_{i\in N}\sum_{j\in N}\int_{\Gamma_{ij}} b_{ij}(t)d(t) = b_T(C_T) \quad \blacksquare\end{aligned}$$

The following lemma will be used to find the minimum dynamic cut.

Lemma 5.2 ([17]) *Let* $\{x_{ij}(t) \ : \ (i,j) \in A, \ t \in [0,T]\}$ *be a set of feasible flows for maximum dynamic flow problem and suppose that the sink node is not s-reachable for any* $t \in [0,T]$. *Define a cut* C_T *by* $C_T(t) = \{i \ : \ i \text{ is } s\text{-reachable at time } t\}$ *for* $t \in [0,T]$. *If the value of the flow is* $V_T(x)$ *and the value of the cut* C_T *is* $b_T(C_T)$ *then* $V_T(x) = b_T(C_T)$.

The termination of the EAF algorithm occurs when the sink node d in the residual network is not s-reachable at any time, i.e. for any conditional augmenting $s-d$ path P, $S_P = \emptyset$. Since for any node $i \in N$, $S_i^{*p,q}$ defines the set of arrival times at the sink when flows from node i reach d, we can use this set to define the minimum dynamic cut. When $S_P = \emptyset$, we define $S_i^{*p,q}(T^{'}), \ T^{'} \leq T$ the set of arrival times at the sink when d is i-reachable but node i is not s-reachable within $T^{'}$, i.e.

$$S_i^{*p,q}(T^{'}) = S_i^{*p,q} \cap [0,T^{'}] \tag{18}$$

Furthermore, we define $\overline{R}_i(T^{'})$ as the possible departure times of flows from node $i \neq s$ to the sink node when the sink is i-reachable but not s-reachable within $T^{'} \leq T$.

$$\overline{R}_i(T^{'}) = \cup_{q=1}^{|Z_i|}\{t - \pi_i^{p,q} \ : \ t \in S_i^{*p,q}(T^{'})\} \tag{19}$$

Hence, $\overline{R}_i(T^{'})$ defines the set of times when node i is in the sink-side of the $s-d$ dynamic cut within time horizon $T^{'}$. We define also the set of times when node i is s-reachable but it is not able to reach the sink node d as the following.

$$R_i(T^{'}) = [0,T^{'}] - \overline{R}_i(T^{'})$$

By using $\overline{R}_i(T^{'})$ and $R_i(T^{'})$, we can formulate $C_{T^{'}}(t)$ and $\Gamma_{ij}(T^{'})$ the as follows.

$$C_{T^{'}}(t) = \{i : t \in R_i(T^{'})\} \tag{20}$$

$$\Gamma_{ij}(T^{'}) = \{ \ t \ : \ t \in R_i(T^{'}), \ t + \lambda_{ij} \notin R_j(T^{'}), \ t + \lambda_{ij} \leq T^{'}\} \tag{21}$$

The complete algorithm is described in detail as follows.

Algorithm 5.1 : Soving CTEAF

Initialization

Determine the distance λ of the shortest path from from s to d.
If $\lambda > T$, then the algorithm terminate with $x_{ij} = 0,\ \forall (i,j) \in A$.
Set all flows $x_{ij} = 0,\ \forall (i,j) \in A$.

SCAP procedure : work with G^R

Step 0 Set $p = 1\ ;\ k_i = 1,\ \forall i \in N\ ;\ S_d^{0,1} = [0,T]\ ;\ \pi_d^{1,1} = 0$
Set $\pi_i^{1,1} := \begin{cases} \lambda_{di} & ,(d,i) \in A_x^R \\ \infty & ,otherwise \end{cases},\ i \in N - \{d\}$
Set $S_i^{1,1} = \{t + \pi_i^{1,1}\ :\ t \in S_{id}\} \cap [0,T]\ ;$
$S_i^{*1,1} = S_i^{1,1} \cap S_d^{0,1}\ ;\ Z_i^1 = \{(d,1)\},\ \forall i \in N$

Step 1 Define $k_j := \mid Z_j^p \mid$ and set $\pi_j^{p+1,q} := \pi_j^{p,q},\ q = 1, \ldots, k_j;$
$\forall j \in N\ ;\ \overline{k}_j := k_j, \forall j \in N$
For $l = 1$ to $\mid A_x^R \mid$ do
If $a_l = (i,j) \in A_x^R$ then
For $q' = 1, \ldots, k_i$ do
$B := \{t + \pi_i^{p,q'} + \lambda_{ij}\ :\ t \in S_{ij}\} \cap [0,T] \cap S_i^{*p,q'}$
If $B \neq \emptyset$ then
If $\exists q \in \{1, \ldots, k_j\} : \pi_j^{p,q} \geq \pi_i^{p,q'} + \lambda_{ij}$ AND $S_j^{*p,q} \subset B$ then
Set $\pi_j^{p+1,q} = \pi_i^{p,q} + \lambda_{ij}\ ;\ S_j^{*p+1,q} = B\ ;\ Z_j^{p+1}(q) = (i,q')$
Else
$\pi_j^{p+1,\overline{k}_j+1} := \pi_j^{p,q'} + \lambda_{ij}\ ;\ S_j^{*p+1,\overline{k}_j+1} := B\ ;$
$Z_j^{p+1}(\overline{k}_j + 1) = (i,q')\ ;\ \overline{k}_j := \overline{k}_j + 1$
$k_j := \overline{k}_j, \forall j \in N$

Step 2 For all $i \in N$ do
If $\exists q, q' \in \{1, \ldots, k_i\} : \pi_i^{p+1,q} \geq \pi_i^{p+1,q'}$ AND $S_i^{*p+1,q} \subset S_i^{*p+1,q'}$ then
Reduce Z_i^{p+1} by deleting $Z_i^{p+1}(q)$ also $S_i^{*p+1,q}$ and $\pi_i^{p+1,q}$, accordingly

Step 3 If $\pi_i^{p,q} = \pi_i^{p+1,q} \forall q = 1, \ldots, k_i\ ;\ \forall i \in N$ then
If $\{\pi_s^{p,q}\ :\ S_s^{*p,q} \neq \emptyset\} \neq \emptyset$ then
$\lambda_s := \min_{1 \leq q \leq k_s} \{\pi_s^{p,q}\ :\ S_s^{*p,q} \neq \emptyset\}$
Find SCAP by using backtracking procedure on Z_i^p with $S_P := S_s^{*p,q*}$ with
$q* := \text{argmin}_{1 \leq q \leq k_s} \{\pi_s^{p,q}\ :\ S_s^{*p,q} \neq \emptyset\}$
Run maximum flow increment procedure.
Else
No conditional augmenting path from source node to sink node.
Determine for all $i \in N : \overline{R}_i := \cup_{q=1}^{|Z_i|} \{t - \pi_i^{p,q}\ :\ t \in S_i^{*p,q}\}$
Run minimum cut procedure.
Else Set p := p+1 and go to Step 1.

Maximum Flow Increment procedure : work with G
Calculate $S_{ij}^P,\ \forall (i,j) \in P$ using Eq. (6).
Calculate maximum incerement flows and get new flows using Eq. (7) - (11).
Revise S_{ij} using(4) and return to SCAP procedure.

Minimum Cut procedure
Define $R_i(T') := [0,T'] - \overline{R}_i(T'),\ \forall i \in N,\ T' \leq T$.
Determine $C_{T'}(t)$ according to Eq. (20) and $\Gamma_{ij}(T')$ according to Eq. (12).
Calculate value of the minimal cut $C(T')$ as in Eq. (13).

6 Illustrative Example

In this section, an example is worked out in detail. The purpose of this section is to clarify the notation and steps of the algorithm. Figure 4 shows the network structure with 6 nodes and 9 arcs of a simple building. Node 0

is the source and node 5 is the safety area. The travel time and arc capacity are attached as attributes on each arc. The fire starts burning room 0 near the exit doors to room 3 in such a way that decreasing the capacity of the arc connecting room 0 and room 3 linearly over time. Finally after 6 time units, arc $(0,3)$ becomes impassable. Capacities of arc $(2,5)$ and $(1,4)$ also decrease linearly over time. Arc $(0,2)$ is some kind of emergency exit which can be used only after five time units from the beginning. The question is how many people can be sent out to the safety within every time $T^{'} \leq T = 12$ units.

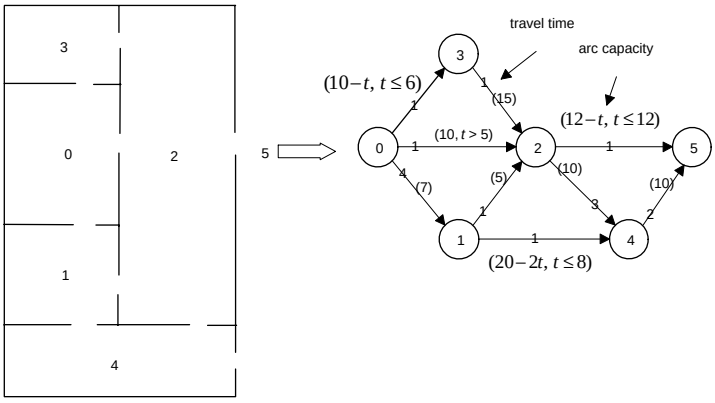

Figure 4: Static network for Example

Table 6 shows the detail calculation of SCAP procedure on iteration 1 with each row show the results after the Step 2.

This procedure results $P = \{0, 2, 5\}$ as the shortest conditional augmenting path with total travel time is 2 time units and $S_P =]7, 12]$. The corresponding distances for node 0 and 2 are 2 and 1. Using Eq. (6), we obtain the interval time when these flows leave node 0 (or entering arc $(0,2)$) and node 2 (or entering arc (2,5)) as follows.
$S^P_{02} =]5, 10]$; $S^P_{25} =]6, 11]$
The residual capacity of each arc $(i,j) \in P$ within S^P_{ij} is
$b^x_{02}(t) = 10,\ t \in]5, 10]$; $b^x_{25}(t) = 12 - t,\ t \in]6, 11]$
The forward translation of these residual capacities give
$b^{'x}_{12}(t) = 10,\ t \in]7, 12]$; $b^{'x}_{26}(t) = 13 - t,\ t \in]7, 12]$
Thus the maximum augmented function along P is obtained as follows.
$\epsilon_P = 13 - t,\ t \in]7, 12]$
The backward translation of ϵ_P to each arc $(i,j) \in P$ is obtained as follows.
$\epsilon_{02}(t) = 11 - t,\ t \in]5, 10]$; $\epsilon_{25}(t) = 12 - t,\ t \in]6, 11]$
Hence, the new dynamic flow distribution is

p	Labels	node 0	node 1	node 2	node 3	node 4	node 5
1	$\pi_i^{p,q}$	∞	∞	1	∞	2	0
	$S_i^{*p,q}$	$\emptyset$	$\emptyset$	]1,12]	$\emptyset$	]2,12]	]0,12]
	Z_i^p	{(5,1)}	{(5,1)}	{(5,1)}	{(5,1)}	{(5,1)}	{(5,1)}
2	$\pi_i^{p,q}$	2	2	1	2	2	0
	$S_i^{*p,q}$	]7,12]	]2,12]	]1,12]	]2,12]	]2,12]	]0,12]
	Z_i^p	{(2,1)}	{(2,1)}	{(5,1)}	{(2,1)}	{(5,1)}	{(5,1)}
3	$\pi_i^{p,q}$	2 ; 6; 3	2	1	2	2	0
	$S_i^{*p,q}$	]7,12] ;]6,12] ;]3,9]	]2,12]	]1,12]	]2,12]	]2,12]	]0,12]
	Z_i^p	{(2,1), (1,1), (3,1)}	{(2,1)}	{(5,1)}	{(2,1)}	{(5,1)}	{(5,1)}
4	$\pi_i^{p,q}$	2 ; 6; 3	2	1	2	2	0
	$S_i^{*p,q}$	]7,12] ;]6,12] ;]3,9]	]2,12]	]1,12]	]2,12]	]2,12]	]0,12]
	Z_i^p	{(2,1), (1,1), (3,1)}	{(2,1)}	{(5,1)}	{(2,1)}	{(5,1)}	{(5,1)}

Table 1: Detail Steps of SCAP Labeling Algorithm of Iteration 1

$x_{02}(t) = 11 - t,\ t \in]5, 10]$; $x_{25}(t) = 12 - t,\ t \in]6, 11]$
and the total flow $x_{50}(t) = 13 - t,\ t \in]7, 12]$

The new residual capacity of each arc is $b_{02}^x(t) = \begin{cases} t-1 & , t \in]5, 10] \\ 10 & , t \in]10, 12] \\ 0 & , \text{ otherwise} \end{cases}$;

$b_{25}^x(t) = \begin{cases} 12-t & , t \in]0, 6] \cup]11, 12] \\ 0 & , \text{ otherwise} \end{cases}$

and keep constant for the other residual capacities. Add negative arcs $(2, 0)$ and $(5, 2)$ with travel time $\lambda_{20}^x = -\lambda_{02}$ and $\lambda_{52}^x = -\lambda_{25}$ and the residual capacities

$b_{20}^x(t) = \begin{cases} 12-t & , t \in]6, 11] \\ 0 & , \text{ otherwise} \end{cases}$; $b_{52}^x(t) = \begin{cases} 13-t & , t \in]7, 12] \\ 0 & , \text{ otherwise} \end{cases}$

Figure 5 shows the new residual network. The new admissible time is

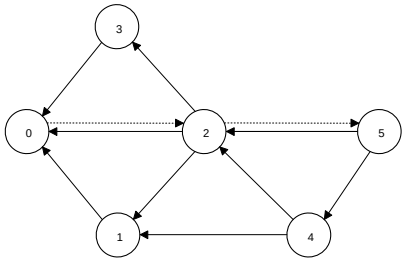

Figure 5: Residual Network $G_x^R = (N, A_x^R)$ after Iteration 1

$S_{02} =]5, 12]$; $S_{20} =]6, 11]$
$S_{25} =]0, 6] \cup]11, 12]$; $S_{52} =]7, 12]$

Figure 6 shows the residual network after iteration 5 and table 6 shows

the calculation of SCAP procedure for iteration 6.

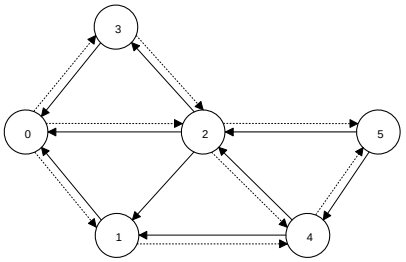

Figure 6: Residual Network $G_x^R = (N, A_x^R)$ after Iteration 5

p	Labels	node 0	node 1	node 2	node 3	node 4	node 5
1	$\pi_i^{p,q}$	∞	∞	1	∞	2	0
	$S_i^{*p,q}$	$\emptyset$	$\emptyset$	]1,3]	$\emptyset$	]2, 11]	]0,12]
	Z_i^p	{(5,1)}	{(5,1)}	{(5,1)}	{(5,1)}	{(5,1)}	{(5,1)}
2	$\pi_i^{p,q}$	∞	3 ; 2	1 ; 5	2	2	0
	$S_i^{*p,q}$	$\emptyset$	]3,9.5] ;]2,3]	]1,3] ;]5,11]	]2,3]	]2,11]	]0,12]
	Z_i^p	{(5,1)}	{(4,1), (2,1) }	{(5,1), (4,1)}	{(2,1)}	{(5,1)}	{(5,1)}
3	$\pi_i^{p,q}$	∞	3 ; 2 ; 6	1; 5	2 ; 6	2	0
	$S_i^{*p,q}$	$\emptyset$	]3,9.5] ;]2,3] ;]7,11]	]1,3] ;]5,11]	]2,3] ;]6,11]	]2,11]	]0,12]
	Z_i^p	{(5,1)}	{(4,1), (2,1), (2,2)}	{(5,1), (4,1)}	{(2,1), (2,2)}	{(5,1)}	{(5,1)}
4	$\pi_i^{p,q}$	∞	3 ; 2 ; 6	1; 5	2 ; 6	2	0
	$S_i^{*p,q}$	$\emptyset$	]3,9.5] ;]2,3] ;]7,11]	]1,3] ;]5,11]	]2,3] ;]6, 11]	]2,11]	]0,12]
	Z_i^p	{(5,1)}	{(4,1), (2,1), (2,2)}	{(5,1), (4,1)}	{(2,1), (2,2)}	{(5,1)}	{(5,1)}

Table 2: Detail Steps of SCAP Labeling Algorithm of Iteration 6

No SCAP is found at iteration 6 and the optimal earliest arrival flow is obtained as follows.

$$x_{01}(t) = \begin{cases} 7 & ,t \in]0, 2.5] \\ 12 - 2t & ,t \in]2.5, 4] \\ 0 & , \text{ otherwise} \end{cases} ; \quad x_{02}(t) = \begin{cases} 10 & ,t \in]5, 6] \\ 11 - t & ,t \in]6, 10] \\ 0 & , \text{ otherwise} \end{cases} ;$$

$$x_{03}(t) = \begin{cases} 10 - t & ,t \in]0, 5] \\ 0 & , \text{ otherwise} \end{cases} ; \quad x_{12}(t) = 0, \ t \in]0, 12] ;$$

$$x_{14}(t) = \begin{cases} 7 & ,t \in]4, 6.5] \\ 20 - 2t & ,t \in]6.5, 8] \\ 0 & , \text{ otherwise} \end{cases} ; \quad x_{24}(t) = \begin{cases} 10 & ,t \in]6, 7] \\ 0 & , \text{ otherwise} \end{cases} ;$$

$$x_{25}(t) = \begin{cases} 12 - t & ,t \in]2, 11] \\ 0 & , \text{ otherwise} \end{cases} ; \quad x_{32}(t) = \begin{cases} 11 - t & ,t \in]1, 6] \\ 0 & , \text{ otherwise} \end{cases} ;$$

$$x_{45}(t) = \begin{cases} 7 & ,t \in]5, 7.5] \\ 22 - 2t & ,t \in]7.5, 9] \\ 10 & ,t \in]9, 10] \\ 0 & , \text{ otherwise} \end{cases} \quad ; \quad x_{50}(t) = \begin{cases} 0 & ,t \in]0, 3] \\ 13 - t & ,t \in]3, 7] \\ 20 - t & ,t \in]7, 9.5] \\ 39 - 3t & ,t \in]9.5, 11] \\ 23 - t & ,t \in]11, 12] \\ 0 & , \text{ otherwise} \end{cases} \quad ;$$

The maximum value $V(T)$ is equal to

$$V(T) = \int_0^{12} x_{50}(t)dt = 85.25$$

The complete solution of the example for the time horizon $T = 12$ is given in the Table 6.

Iteration No.	SCAP	S_P
1	$0 \to 2 \to 5$ $\lambda_0 = 2; \lambda_2 = 1$	]7,12]
2	$0 \to 3 \to 2 \to 5$ $\lambda_0 = 3; \lambda_3 = 2; \lambda_2 = 1$	$]3, 7]$
3	$0 \to 2 \to 4 \to 5$ $\lambda_0 = 6; \lambda_2 = 5; \lambda_4 = 2$	$]11, 12]$
4	$0 \to 1 \to 4 \to 5$ $\lambda_0 = 7; \lambda_1 = 3; \lambda_4 = 2$	$]7, 11]$
5	$0 \to 3 \to 2 \to 4 \to 5$ $\lambda_0 = 7; \lambda_3 = 6; \lambda_2 = 5; \lambda_4 = 2$	$]11, 12]$
$\overline{R}_0 = \emptyset; \overline{R}_1 =]0, 6.5]; \overline{R}_2 =]0, 6]; \overline{R}_3 =]0, 5]; \overline{R}_4 =]0, 9]; \overline{R}_5 =]0, 12]$		
$\Gamma_{01} =]0, 2.5]; \Gamma_{02} = \emptyset; \Gamma_{03} =]0, 4]; \Gamma_{12} = \emptyset; \Gamma_{14} =]6.5, 8]$ $\Gamma_{24} = \emptyset; \Gamma_{25} =]6, 11]; \Gamma_{32} = \emptyset; \Gamma_{45} =]9, 10];$		

Table 3: Solution of the Illustrative Numerical Example

The minimum dynamic cut is obtained from $S_i^{*p,q}$ of the last iteration (iteration 6), by using Eq. (19) - (20) and Eq. (12). The source side of the cut at any time t is obtained as follows.

$$C_T(t) := \begin{cases} \{0\} & ,t \in]0, 5] \\ \{0, 3\} & ,t \in]5, 6] \\ \{0, 3, 2\} & ,t \in]6, 6.5] \\ \{0, 3, 2, 1\} & ,t \in]6.5, 9] \\ \{0, 3, 2, 1, 4\} & ,t \in]9, 12] \end{cases}$$

Using Eq. (13), the minimal cut value is obtained equal to 85.25 units which is equal to $V(T)$.

7 Solving Evacuation Problems with Known Occupancy

For a given T the building is cleared within the time horizon T if and only if the corresponding dynamic network allows a maximal $s-d$ flow with flow value at least as large as the initial occupancy q. In this section we show how to minimize T. The interrelation between EAF and the quickest flow problem defined in the introduction is explained by the following theorem.

Theorem 7.1 *Let $V_T(x)$ be the total value of flow x entering the sink within time horizon T, i.e. $V_T(x) = \sum_{(i,d)\in A} \int_0^T x_{id}(t)dt$. Suppose x is the earliest arrival flow. If $T^* := min\ \{T' \ :\ V_{T'}(x) \geq q,\ T' \leq T\}$ exists then T^* solves the quickest flow problem with initial ocupancy q.*

Proof :
If T^* does not solve the quickest flow problem then there is $T'' < T^*$ and flow x'' with value $V_{T''}(x'') = q$. Hence $V_{T''}(x) < V_{T''}(x'')$. This contradicts that x is also maximum at $T'' < T$. ■

The theorem suggests a simple algorithm to solve quickest flow problem via EAF.

Algorithm 7.1 : Solving Continuous Time QFP

Step 0	Set the time T large enough such that the problem is feasible, i.e. if x is the maximum dynamic flow, then $V_T(x) \geq q$
Step 1	Connect source node s to supersource node s' with capacity $b_{s's}(t) = q,\ t \in [0,T]$
Step 2	Find x the solution of the continuous time earliest arrival flow problem with T time periods.
Step 3	Find $T^* := \min\{T' \ :\ V_{T'}(x) \geq q,\ T' \leq T\}$ T^* is the solution of the quickest flow problem.

Example 7.1
The minimum time to clear the network with 50 initial occupancies at node 0 is $T^* = 8.467$ with the following optimal flows distribution.

$$x_{01}(t) = \begin{cases} 7 & ,t \in]0, 1.467] \\ 0 & ,\ \text{otherwise} \end{cases} \ ;\ x_{02}(t) = \begin{cases} 11-t & ,t \in]5, 6.467] \\ 0 & ,\ \text{otherwise} \end{cases} \ ;$$

$$x_{03}(t) = \begin{cases} 10-t & ,t \in]0, 4] \\ 0 & ,\ \text{otherwise} \end{cases} \ ;\ x_{12}(t) = 0,\ t \in]0, 8.467] \ ;$$

$$x_{14}(t) = \begin{cases} 7 & ,t \in]4, 5.467] \\ 0 & ,\ \text{otherwise} \end{cases} \ ;\ x_{24}(t) = 0,\ t \in]0, 8.467] \ ;$$

$$x_{25}(t) = \begin{cases} 12-t & ,t \in]2, 7.467] \\ 0 & , \text{ otherwise} \end{cases} \; ; \; x_{32}(t) = \begin{cases} 11-t & ,t \in]1, 5] \\ 0 & , \text{ otherwise} \end{cases} \; ;$$

$$x_{45}(t) = \begin{cases} 7 & ,t \in]5, 6.467] \\ 0 & , \text{ otherwise} \end{cases} \; ; \; x_{50}(t) = \begin{cases} 0 & ,t \in]0, 3] \\ 13-t & ,t \in]3, 7] \\ 20-t & ,t \in]7, 8.467] \end{cases} \; ;$$

Some calculations required to determine the minimum dynamic cut within $T^* = 8.467$ is summarized in Table 4.

Table 4: Some Calculations Required to Determine the Minimum Dynamic Cut within $T^* = 8.467$

	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5
$\overline{R}$	$\emptyset$	$]0, 5.467]$	$]0, 3.467]$	$]0, 2.467]$	$]0, 6.467]$	$]0, 8.467]$
$C_{T^*}(t) := \begin{cases} \{0\} & ,t \in]0, 2.467] \\ \{0,3\} & ,t \in]2.467, 3.467] \\ \{0,3,2\} & ,t \in]3.467, 5.467] \\ \{0,3,2,1\} & ,t \in]5.467, 6.467] \\ \{0,3,2,1,4\} & ,t \in]6.467, 8.467] \end{cases}$						
$\Gamma_{01} =]0, 1.467]; \Gamma_{02} = \emptyset; \Gamma_{03} =]0, 1.467]; \Gamma_{12} = \emptyset; \Gamma_{14} = \emptyset$ $\Gamma_{24} = \emptyset; \Gamma_{25} =]3.467, 7.467]; \Gamma_{45} = \emptyset$						
Minimum value of the dynamic cut $(C_{8.467}, \overline{C}_{8.467})$ is 50 units.						

References

[1] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B., Network Flows : Theory, Algorithms, and applications, Prentice Hall, Englewood Cliffs, New Jersey (1993).

[2] Aronson, Jay E., A Survey of Dynamic Network Flows, *Annals of Operation Research*, 20 : 1-66 (1989).

[3] Burkard, R.E., Dlaska, K., and Klinz, B., The Quikest Flow Problem, *ZOR-Methods and Models of Operations Research*, 37 : 31-58 (1993).

[4] Chalmet, L.G., Francis, R.L., and Saunders, P.B., Network Models for Building Evacuation, *Management science*, 28 : 86-105 (1982).

[5] Choi, W., Francis, R.L., Hamacher, H.W., and Tufekci, S., Network Models of Building Evacuation Problems With Flow-Dependent Exit Capacities, *Operational Research*, 1047-1059 (1984).

[6] Choi, W., Francis, R.L., Hamacher, H.W., and Tufekci, S., Modelling of Building Evacuation Problems with Side Constraints, *European Journal of Operation Research*, 35 : 98-110 (1988).

[7] Fleischer, Lisa, Faster Algorithms for the Quickest Transshipment Problem, *Proceedings of 9th Annual ACM-SIAM Symposium on Discrete Algorithms* 147-156 (1998).

[8] Hamacher, H.W. and Foulds, L.R., Algorithms for Flows with Parametric Capacities, *ZOR - Methods and Models of Operations Research*, 33 : 21-37 (1989).

[9] Hamacher, H.W., Tufekci, S., On the Use of Lexicographic Min Cost Flows in Evacuation Modeling, *Naval Research Logistics*, 34 : 487-503 (1987).

[10] Hoppe, B. and Tardos, E., Polinomial Time Algorithms for Some Evacuation Problems, *Proc. of 5th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 433-441 (1994).

[11] Hoppe, B. and Tardos, E., The Quickest Transshipment Problem, *Proc. of 6th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 512-521 (1995).

[12] Jarvis, J.J. and Ratliff, H.D., Some Equivalent Objectives for Dynamic Network Flow Problems, *Management science*, 28 : 106-108 (1982).

[13] Kostreva, M.M., and Wiecek, M.M., Time Dependency In Multiple Objective Dynamic Programming, *Journal of mathematical Analysis and Application*, 173(1) : 289-307 (1993).

[14] Kostreva, Michael M., Mathematical Modeling of Human Egress from Fires in Residential Buildings, *Building and Research Laboratory of the National Institute of Standards and Technology* , Technical Report, NIST-GCR-94-643 (1994).

[15] Minieka, E., Maximal, Lexicographic, and Dynamic Network Flows, *Operations Research*, 21 : 517-527 (1973).

[16] Montes, Christian, Evacuation of Buildings, M.Sc. Thesis, *Department of Mathematics, Universität Kaiserslautern*, Kaiserslautern, Germany [1994].

[17] Philpott, A.B., Continuous-Time Flows in Networks, *Mathematics of Operation Research*, 15(4) : 640-661 (1990).