

Theoretical and Practical Issues of Evacuation Planning in Urban Areas

Naoyuki KAMIYAMA, Naoki KATOH, Atsushi TAKIZAWA

Abstract—In the evacuation problem by using dynamic network flow, the time-expanded network introduced by Ford and Fulkerson [6] has been used in many applications. Although the algorithm based on the time-expanded network can be easily implemented, the size of the time-expanded network depends on granularity of the unit time adopted. In this paper, we present the results of the trade-off between actual runtime and the accuracy of model through numerical examples arising from real data. Furthermore, we consider several other features from the viewpoint of actual evacuation planning: (i) To avoid disorderly congestion in evacuation situation, blocking at intermediate vertices is undesirable. (ii) It is preferable that all the supplies at the same vertex take the same path to a sink. In this paper, we present the well-defined theoretical models in order to solve the above problems, and the results concerning these models.

Index Terms—Dynamic network flow, Evacuation problem, Combinatorial optimization

I. INTRODUCTION

IN December 2004, the Sumatra-Andaman earthquake occurred. It triggered tsunamis, and tragedy fell upon many people. Not only earthquakes but also diverse disasters occurred and caused serious damages in many countries. Therefore it is very important to establish crisis management systems against large-scale disasters such as big earthquakes, conflagrations and tsunamis to secure evacuation pathways and to effectively guide residents to a safe place. The problem for finding the most effective plan to evacuate people to safe place has been modelled as an *evacuation problem* by using *dynamic network flow*. In the evacuation problem, we are given a directed graph $D = (V, A)$ which consists of a vertex set V with supply $b(v)$ on every vertex v and an arc set A with capacity $c(e)$ and transit time $\tau(e)$ on every arc e and a single sink $s \in V$. If we consider urban evacuation, vertices model buildings, rooms, exits and so on, and arcs model pathways or roads. For an arc e , capacity $c(e)$ represents the number of people which can traverse e per unit time, and transit time $\tau(e)$ represents the time required to traverse e . For any vertex v , supply $b(v)$ represents the number of people which exist at v . The evacuation problem asks to find the minimum time required to send all the supplies to a sink.

The first algorithm to solve the dynamic flow problem was proposed by Ford and Fulkerson [6]. It first transforms the problem to a maximum-flow problem whose size is the original network size times the time horizon T , and thus its running time is pseudo-polynomial. Theoretically, the first polynomial time algorithm for the evacuation problem was proposed by Hoppe and Tardos [8]. However it requires to use the submodular function minimization as a subroutine. Their algorithm requires

The authors are with Department of Architecture and Architectural Engineering, Kyoto University, Kyotodaigaku-Katsura, Nishikyo-ku, Kyoto, 615-8540, Japan. E-mail: {is.kamiyama, naoki, kukure}@archi.kyoto-u.ac.jp

$O(\log(n \cdot \max_{e \in A} \tau(e) + \sum_{v \in V} b(v)))$ submodular function minimizations. Hence the running time is high-order polynomial, and the algorithm is not practical in general. Therefore it is necessary to devise a faster algorithm for a tractable and practically useful subclass of this problem. In our previous papers [10] and [11], we have presented respectively an $O(n \log n)$ time algorithm for a $\sqrt{n} \times \sqrt{n}$ grid network with uniform transit time and uniform arc capacity where n denotes the number of vertices in the given network and the algorithm for larger class of networks including the grid network with uniform transit time and uniform arc capacity.

When we consider the evacuation planning in practical situations (e.g. buildings, cities, and so on), the algorithm of [8] is very complicated, and it is hard to implement. Furthermore, in general the network which we need to treat may not belong to the above special classes studied by [10] and [11]. Therefore, in many applications we need to use the time-expanded network introduced by Ford and Fulkerson [6]. Although the algorithm based on the time-expanded network can be easily implemented, the size of the time-expanded network depends on granularity of the *unit time* adopted, e.g., if we set the unit time to one second, the size of the time-expanded network is sixty times larger than the one in the case where the unit time is set to one minute. Namely, a more accurate model relies on finer granularity, requiring longer running time. We need to make decision considering the trade-off between actual runtime and the accuracy of model. In our paper, we present the results concerning this trade-off through numerical examples arising from real data.

Moreover we need to consider several other features from the viewpoint of actual evacuation planning.

- i. The first problem is *blocking* at intermediate vertices. To avoid disorderly congestion in evacuation situation, blocking at intermediate vertices is undesirable. Theoretically, it is known [5] that there exists an optimal flow without blocking for the evacuation problem. However, such optimal flow incurs a situation such that people at sources must wait even if the outgoing arc is not occupied. Thus, this flow is not “psychologically” acceptable for human. In this paper, we present a new model for a flow without blocking.
- ii. The commodity which flows through the network represents people. Therefore, it is desirable for the evacuation guidance that all the supplies at the same vertex take the same path to a sink. In this paper, we present the *evacuation problem* which is the evacuation problem under the constraint such that all the supplies at the same vertex take the same path to a sink.

These practical issues are not only very interesting from practical viewpoint but provide new theoretical issues. In our paper, we present the well-defined theoretical models in order to solve the above problems, and the results concerning these models.

This paper is organized as follow. Section II introduce notations

and fundamental definitions. Section III presents the results of the trade-off between actual runtime and the accuracy of model through numerical examples arising from real data. Section IV consider a new model for a flow without blocking. In Section V, we consider the evacuation problem under the constraint such that all the supplies at the same vertex take the same path to a sink. Section VI concludes this paper.

II. PRELIMINARIES

Let \mathbb{R}_+ and \mathbb{Z}_+ denote the set of nonnegative reals and non-negative integers, respectively. We will not distinguish between a singleton $\{x\}$ and its element x . For any finite set X , we define $|X|$ as the number of elements that belong to X .

A. Directed graph

Let $D = (V, A)$ be a directed graph. We denote by $e = uv$ an arc e whose tail is u and head is v . If $e = uv$ has no parallel arc, we may simply write uv . For $X \subseteq V$, we define $\delta^+(X) = \{e = xy \in A: x \in X, y \in V - X\}$ and $\delta^-(X) = \{e = xy: x \in V - X, y \in X\}$. Throughout this paper, n and m denote $|V|$ and $|A|$, respectively.

B. Dynamic network

Let $\mathcal{N} = (D = (V, A), c, \tau, b, s)$ be a dynamic network \mathcal{N} which consists of a directed graph $D = (V, A)$, a capacity function $c: A \rightarrow \mathbb{R}_+$ which represents the upper bound for the rate of flow that enters an arc per unit time, a transit time function $\tau: A \rightarrow \mathbb{Z}_+$ which represents the time required to traverse an arc, a supply function $b: V \rightarrow \mathbb{R}_+$ which represents the supply of a vertex, and a single sink $s \in V$. Since we consider evacuation to s , we assume that s has no leaving arcs and no supply, and any vertex is reachable to s . We define a *length* of a path p in D as $\sum_{e \in p} \tau(e)$. We define a *dynamic network flow* $f: A \times \mathbb{Z}_+ \rightarrow \mathbb{R}_+$ in \mathcal{N} as follows. For $e \in A$ and $\theta \in \mathbb{Z}_+$, we denote by $f(e, \theta)$ the flow rate entering e at the time step θ which arrives at the head of e at the time step $\theta + \tau(e)$. We call f *feasible dynamic network flow* in \mathcal{N} if it satisfies the following three conditions, i.e., *capacity constraint*, *flow conservation*, and *demand constraint* [13].

Capacity constraint: For any $e \in A$ and $\theta \in \mathbb{Z}_+$,

$$0 \leq f(e, \theta) \leq c(e).$$

Flow conservation: For any $v \in V$ and $\Theta \in \mathbb{Z}_+$,

$$\sum_{e \in \delta^+(v)} \sum_{\theta=0}^{\Theta} f(e, \theta) - \sum_{e \in \delta^-(v)} \sum_{\theta=0}^{\Theta - \tau(e)} f(e, \theta) \leq b(v). \quad (1)$$

Demand constraint: There exists $\Theta \in \mathbb{Z}_+$ such that

$$\sum_{e \in \delta^-(s)} \sum_{\theta=0}^{\Theta - \tau(e)} f(e, \theta) = \sum_{v \in V} b(v). \quad (2)$$

Notice that (1) allows blocking at vertices. Let $\mathcal{F}(\mathcal{N})$ be the set of all feasible dynamic network flows. For $f \in \mathcal{F}(\mathcal{N})$, let $\Theta(f)$ denote the minimum time step Θ satisfying (2). The *evacuation problem* asks to find the minimum value of $\Theta(f)$ among all $f \in \mathcal{F}(\mathcal{N})$. Given a dynamic network \mathcal{N} , the evacuation problem $\text{EP}(\mathcal{N})$ is formally defined as follows:

$$\text{EP}(\mathcal{N}): \text{minimize } \{\Theta(f): f \in \mathcal{F}(\mathcal{N})\}.$$

Given time horizon T , we define the *decision version of EP*(\mathcal{N}) *with time horizon T* as the problem which determines whether there exists a feasible dynamic network flow f with $\Theta(f) \leq T$ in \mathcal{N} .

Throughout this paper, we use the convention that in a figure of a dynamic network, the pair of numbers attached to an arc indicates the capacity and the transit time. Moreover, the number attached to a vertex indicates the supply (see Figure 1(a) for example).

C. Static network

Let $\hat{\mathcal{N}} = (\hat{D} = (\hat{V}, \hat{A}), \hat{c}, \hat{b}, \hat{S})$ be a *static network*¹ $\hat{\mathcal{N}}$ which consists of a directed graph $\hat{D} = (\hat{V}, \hat{A})$, a capacity function $\hat{c}: \hat{A} \rightarrow \mathbb{R}_+$, a supply function $\hat{b}: \hat{V} \rightarrow \mathbb{R}_+$, and a set of sinks $\hat{S} \subseteq \hat{V}$. We call $f: \hat{A} \rightarrow \mathbb{R}_+$ a *feasible static network flow* in $\hat{\mathcal{N}}$ if it satisfies the following two conditions, i.e., *capacity constraint* and *flow conservation*.

Capacity constraint: For any $e \in \hat{A}$,

$$0 \leq f(e) \leq \hat{c}(e).$$

Flow conservation: For any $v \in \hat{V} - \hat{S}$,

$$\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = \hat{b}(v).$$

If there exists a feasible flow in $\hat{\mathcal{N}}$, $\hat{\mathcal{N}}$ is called *feasible*.

Throughout this paper, in a figure of a static network, the number attached to an arc and a vertex indicates the capacity and the supply, respectively. If no number is attached to an arc, the capacity of this arc is infinite. Moreover, if no number is attached to a vertex, the supply of this vertex is zero.

D. Time-expanded network

In order to solve the decision version of $\text{EP}(\mathcal{N})$ with time horizon T , Ford and Fulkerson [6] introduced the *time-expanded network* $\mathcal{N}(T)$ which is a static network such that for any $v \in V$ and $i = 0, 1, \dots, T$, there is a vertex v_i , and for any $e = uv \in A$ and $i = 0, 1, \dots, T - \tau(e)$, there is an arc $e_i = u_i v_{i+\tau(e)}$ whose capacity is $c(e)$, and for any $v \in V$ and $i = 0, 1, \dots, T - 1$, there is a *holdover arc* $v_i v_{i+1}$ with infinite capacity. For every $v \in V$, the supply of v_0 is set to $b(v)$ and the supplies of all the other vertices v_i for $i = 1, \dots, T$ are set to zero. Let s_T be a single sink of $\mathcal{N}(T)$ (see Figure 1).

It is known [6] that there exists a feasible dynamic flow f with $\Theta(f) \leq T$ if and only if $\mathcal{N}(T)$ is feasible. Although we can decide whether the time-expanded network is feasible or not by solving the maximum-flow problem, the running time is pseudo-polynomial because the size of the time-expanded network is proportioned to T .

E. Grid network

In this paper, we often consider the evacuation problem defined on a *grid graph* (see Figure 2) since the grid structure often appears in modelling building corridors and city streets.

Here we give the formal definition of a *grid graph* $D = (V, A)$. We assume any $v \in V$ is on grid points $\{0, 1, \dots, N\} \times$

¹In order to distinguish classical network from dynamic network, we call classical network *static network*, and flow defined on static network is called *static flow* or simply *flow*.

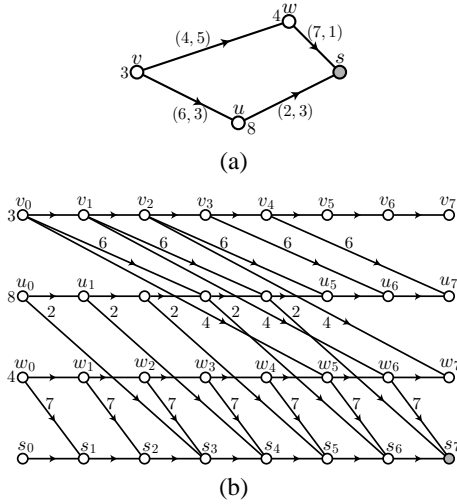


Fig. 1. (a) Dynamic network \mathcal{N} . (b) Time-expanded network $\mathcal{N}(7)$.

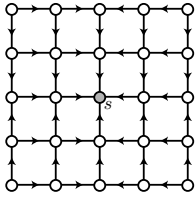


Fig. 2. Grid graph.

$\{0, 1, \dots, N\}$ in \mathbb{R}^2 , and a vertex is identified with (x, y) with $0 \leq x \leq N$ and $0 \leq y \leq N$. The distance between two vertices (x, y) and (x', y') is defined as $|x - x'| + |y - y'|$. Two vertices v and w are connected by an arc if and only if the distance between v and w is equal to one. Given a specified vertex $s \in V$ as a sink, the arc which connects v and w is directed from v to w if and only if the distance from w to s is smaller than that from v to s . A dynamic network defined on a grid graph is called a *grid network*.

III. TRADE-OFF BETWEEN ACTUAL RUNTIME AND ACCURACY OF EVACUATION MODEL

In this section, we present the results of the trade-off between actual runtime and the accuracy of model through numerical examples arising from real data. As was shown in Section II-D, we can solve the decision version of $\text{EP}(\mathcal{N})$ by solving maximum-flow problem defined on the time-expanded network. Thus, we can solve $\text{EP}(\mathcal{N})$ in the framework of binary search. To apply the binary search method, we first have to estimate the upper bound for the optimal value of $\text{EP}(\mathcal{N})$. It is known [8] that the optimal value of $\text{EP}(\mathcal{N})$ is bounded by $n \cdot \max_{e \in A} \tau(e) + \sum_{v \in V} b(v)$.

A. Modelling Kyoto City

In this paper, we model *Kyoto city* as an input data. This city has a grid structure. In this city, it is planned that people who live in a square of side length of two kilometers evacuate to the same refuse such as a schoolyard. Thus, we consider the evacuation problem defined on 20×20 grid network and we regard the distance between the adjacent vertices as one hundred meters. If we regard one time unit of the evacuation problem as one second,

the transit time of an arc becomes about 150 time units where we assume that human can walk about 65 centimeters in one second. Next we consider supplies of vertices. Since about 150 persons live in a square of side length of one hundred meters, a supply of a single vertex is about 150 if we regard one supply as one person. If we regard one time unit and one supply as one second and one person respectively, the upper bound of the optimal value of $\text{EP}(\mathcal{N})$, i.e., $n \cdot \max_{e \in A} \tau(e) + \sum_{v \in V} b(v)$ is more than a hundred thousand, and the size of a time-expanded network becomes huge. Thus, in our numerical examples, we regard one supply as four persons, and one unit time as the following four cases (i) five seconds, (ii) fifteen seconds, (iii) twenty-five seconds, and (iv) fifty seconds. In the following subsection, we present the time spent to solve the evacuation problem and the accuracy of each model.

B. Numerical Example

First we randomly generate four sets of problem instances each of which consists of nine dynamic networks with the following properties:

- 20×20 grid network.
- a single sink is randomly placed in a grid network.
- supplies of vertices are between 25 and 45.
- capacities of arcs are between 1 and 10.
- transit times of arcs are between 100 and 200 seconds.

Four sets of problem instances are generated by setting the unit time of transit time to five, fifteen, twenty-five and fifty seconds, respectively. These four sets are denoted by Cases (i), (ii), (iii) and (iv), respectively. These experiments were done on a PC with a Anthlon 64, 2.20 GHz with 1.00 GB memory, and the code was written by C++. Furthermore, We used LEDA [1] to implement the maximum-flow algorithm. Table I shows the results of this experiment. In this table, “opt” represents the optimal value, and “time” represents the runtime in seconds spent to solve the problem.

TABLE I
RESULTS OF EXPERIMENT.

No.	Case (i)		Case (ii)		Case (iii)		Case (iv)	
	opt	time	opt	time	opt	time	opt	time
1	930	624	311	72	187	23	94	8
2	2779	899	1109	133	693	55	347	14
3	5073	12012	2027	507	1266	284	633	73
4	998	1341	341	132	207	47	104	15
5	790	864	297	56	194	27	98	7
6	684	536	240	69	146	28	79	7
7	2248	1988	842	164	517	74	259	23
8	608	420	212	40	129	20	69	5
9	888	1237	322	113	197	50	101	11
average	1666	2213	633	142	392	68	198	18

With respect to the time spent to solve the problem, the average runtime in Case (i) was sixteen times longer than that in Case (ii), although the unit time of Case (i) is three times longer than that of Case (ii). Similarly, for Cases (ii), (iii) and (iv), the average runtime decreases according to granularity. On the other hand, for the optimal value, there was a drastic difference between Case (i) and Case (ii). In these cases, the average of the evacuation time in a real world are 8330 seconds and 9495 seconds, respectively. The difference between two cases is equal to about seventeen

minutes. That is, the granularity of Case (ii) is too rough. From this experiment, we can see that the time unit is set to at most five seconds although the time spent to solve the problem is longer than that in the other cases.

IV. SMOOTH EVACUATION

In this section, we consider a dynamic network flow without blocking. To avoid disorderly congestion in evacuation situation, blocking at intermediate vertices is undesirable. Theoretically, it is known [5] that there exists an optimal flow without blocking for the evacuation problem. However, this optimal flow without blocking incurs a situation such that people at sources wait even if the outgoing arc is not occupied. Thus, this flow is not *psychologically* acceptable for human.

Let us consider a small example (see Figure 3). First we consider the case where supplies at each vertex enter into an incident arc as much as possible. Table (3) shows the amount of flow entering into each arc in this case. For convenience, the supply of v and that of w are assumed to be distinguishable. f_v and f_w represent the amount of flow of the supply of v and w , respectively. In this case, the supply of w has to wait at v . Thus, this flow is not acceptable. Next we consider a flow as shown in Table (4). In this flow, although no supply waits at intermediate vertex, the supply of v yield to the supply of w , and it is unnatural in an evacuation situation. Finally we consider a flow as shown in (5). In this flow, the supplies of w do not have to wait at v . However, at time step 0, the supplies of w wait although an arc wv is empty.

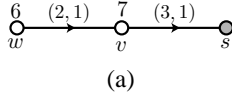


Fig. 3. Input dynamic network.

time		0	1	2	3	4
vs	f_v	3	3	1	0	0
	f_w	0	0	2	3	1
wv	f_w	2	2	2	0	0

(3)

time		0	1	2	3	4
vs	f_v	3	1	1	1	1
	f_w	0	2	2	2	0
wv	f_w	2	2	2	0	0

(4)

time		0	1	2	3	4
vs	f_v	3	3	1	0	0
	f_w	0	0	2	2	2
wv	f_w	0	2	2	2	0

(5)

From the above discussion, we define a *smooth evacuation* which is a *natural* dynamic flow without blocking as follows. Intuitively speaking, a smooth evacuation satisfies the following two properties:

- (i) Blocking at intermediate vertices is not allowed.
- (ii) For each vertex v and time step θ , the number of supplies of v going out from v at time step θ is the minimum of $\sum_{e \in \delta^+(v)} c(e)$ and the supplies remaining at v .

The property (i) means that people can “smoothly” evacuate without any intermediate blocking. The property (ii) means that people do not yield to those from behind.

Here we give a formal definition of a smooth evacuation. For $v \in V$, let $c(v) = \sum_{e \in \delta^+(v)} c(e)$. We define $g: V \times \mathbb{Z}_+ \rightarrow \mathbb{R}_+$ as follows:

$$g(v, \theta) = \begin{cases} c(v), & \text{if } \theta \leq t^*(v) \\ b(v) - t^*(v) \cdot c(v), & \text{if } \theta = t^*(v) + 1 \\ 0, & \text{if } \theta > t^*(v) + 1 \end{cases}$$

where

$$t^*(v) = \left\lfloor \frac{b(v)}{c(v)} \right\rfloor.$$

A smooth evacuation is a feasible dynamic network flow f which satisfies the following property (1) for every $v \in V - s$ and $\theta \in \mathbb{Z}_+$ instead of (1).

$$\sum_{e \in \delta^+(v)} f(e, \theta) - \sum_{e \in \delta^-(v)} f(e, \theta - \tau(e)) = g(v, \theta). \quad (6)$$

In this section, we give the necessary and sufficient condition such that there exists a smooth evacuation in a dynamic network with *uniform path-lengths*. The uniform path-lengths property is that for each vertex $v \neq s$, the sum of transit times of arcs on any path from v to s takes the same value. The class of a dynamic network with uniform path lengths include a tree, a grid network with uniform transit time.

From here, we assume that \mathcal{N} is a dynamic network with uniform path-lengths. Here we introduce necessary notations. For $W \subseteq V$, let $D[W]$ denote the directed subgraph of D induced by W . For each $v \in V$, we define l_v as the length of a path from v to s . Let us arrange the distinct values in $\{l_v: v \in V\}$ as $L_1 < \dots < L_k$ where $L_1 = 0$ and k is the number of the distinct path-lengths to s in \mathcal{N} , i.e., $|\{l_v: v \in V\}|$. We say a vertex v is at level i when $l_v = L_i$, which is denoted by $lev(v) = i$. Letting $T \in \mathbb{Z}_+$ be a sufficient large number which is more than the optimal value of $EP(\mathcal{N})$ and $L_{k+1} = T + 1$, we partition interval $[0, T]$ into I_1, I_2, \dots, I_k such that $I_i = [L_i, L_{i+1} - 1]$ for $i = 1, \dots, k$. For $i = 1, \dots, k$, let $V_{\leq i} = \{v \in V: lev(v) \leq i\}$. For example, for \mathcal{N} with $T = 7$ in Figure 1(a), we obtain $(l_s, l_w, l_u, l_v) = (0, 1, 3, 6)$. Thus, we have $k = 4$ and $I_1 = \{0\}, I_2 = \{1, 2\}, I_3 = \{3, 4, 5\}, I_4 = \{6, 7\}$.

Theorem 1 There exists a smooth evacuation in a dynamic network with uniform path-lengths \mathcal{N} if and only if for every $i = 1, \dots, k$, a static network $\mathcal{N}_i = (D[V_{\leq i}], c_i, b_i, s)$ is feasible where c_i is a restriction of c on arcs in $D[V_{\leq i}]$ and b_i is defined for $v \in V_{\leq i}$ as follows:

$$b_i(v) = g(v, L_i - l_v).$$

Proof: We first remark that $g(v, L_i - l_v)$ represents the amount of the supply of v which has to reach s at time step L_i for each $i = 1, \dots, k$ and $v \in V_{\leq i}$. Since l_v is the length of a path from v to s and a smooth evacuation does not allow blocking, the supply of v which starts from v at time step $L_i - l_v$ reaches to s at time step L_i .

To prove the theorem, we first show an important property concerning the time-expanded network $\mathcal{N}(T)$ of a dynamic network with uniform path-lengths. As was shown in [7] and [11], the time-expanded network of a dynamic network with uniform path-lengths has *layered structure*. More formally,

- (L0) $\mathcal{N}(T)$ consists of T components such that for any $\theta \in \{0, 1, \dots, T\}$, the θ -th component is isomorphic to $D[V_{\leq i}]$

where $\theta \in I_i$. And, the capacity of an arc e in the θ -th component is equal to $c(e)$.

(L1) Consecutive components are connected by holdover arcs.

For example, the time-expanded network of \mathcal{N} is illustrated in Figure 1(a). That is, the static network in Figure 4 is equal to that in Figure 1(b) (notice that we remove vertices which are not reachable to s and arcs incident to them from the network in Figure 1(b)).

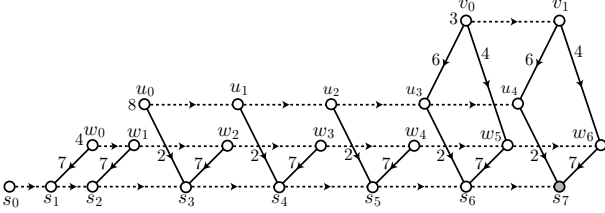


Fig. 4. Layered structure of $\mathcal{N}(T)$.

Thus, a smooth evacuation in \mathcal{N} is equivalent to a feasible static flow f in a static network $\tilde{\mathcal{N}}(T)$ obtained by transforming $\mathcal{N}(T)$ as follows:

- Remove holdover arcs in $\{v_i v_{i+1} : v \in V - s, i = 1, \dots, T-1\}$ from $\mathcal{N}(T)$ (see Figure 5).
- Change the supply of v from $b(v)$ to $g(v, \theta)$ for $v \in V$ and $\theta = 0, \dots, T$ (see Figure 5).

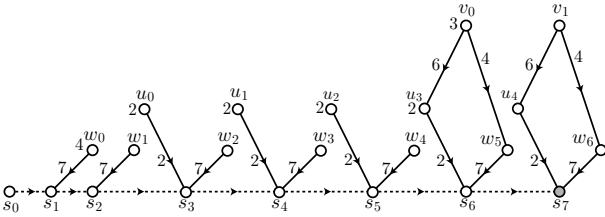


Fig. 5. $\tilde{\mathcal{N}}(T)$ for $\mathcal{N}(7)$ in Figure 4.

We denote by \tilde{D} and \tilde{b} the underlying graph and the supply function of $\tilde{\mathcal{N}}(T)$, respectively. For $\theta = 1, \dots, T$, let \tilde{D}_θ be the underlying graph of the θ -th component. For example, in Figure 5, $\tilde{D}_4 = \tilde{D}[\{s_4, w_3, u_2\}]$. Furthermore, for $\theta = 1, \dots, T$ let $\tilde{\mathcal{N}}_\theta = (\tilde{D}_\theta, \tilde{c}_\theta, \tilde{b}_\theta, s_\theta)$ where \tilde{c}_θ and \tilde{b}_θ are restriction of c and \tilde{b} on \tilde{D}_θ . Since (L0) clearly holds for $\tilde{\mathcal{N}}(T)$ and there exist holdover arcs only between copies of s , $\tilde{\mathcal{N}}(T)$ is feasible if and only if $\tilde{\mathcal{N}}_\theta$ is feasible for every $\theta = 1, \dots, T$. For example, in $\tilde{\mathcal{N}}(7)$ of Figure 5, we consider the networks $\tilde{\mathcal{N}}_0, \tilde{\mathcal{N}}_1, \dots, \tilde{\mathcal{N}}_7$, separately whose underlying graphs are $\tilde{D}[s_0]$, $\tilde{D}[\{s_1, w_0\}]$, $\tilde{D}[\{s_2, w_1\}]$, $\tilde{D}[\{s_3, w_2, u_0\}]$, $\tilde{D}[\{s_4, w_3, u_1\}]$, $\tilde{D}[\{s_5, w_4, u_2\}]$, $\tilde{D}[\{s_6, w_5, u_3, v_0\}]$, $\tilde{D}[\{s_7, w_6, u_4, v_1\}]$.

Since $\tilde{b}(v_\theta)$ is non-increasing in θ and the underlying graph of the θ -th component is isomorphic to $D[V_{\leq i}]$ for $i = 1, \dots, k$ with $\theta \in I_i$ from (L0) (see Figure 4), if there exist feasible flows in $\tilde{\mathcal{N}}_{L_i}$ for all $i = 1, \dots, k$, there exist feasible flows in $\tilde{\mathcal{N}}_\theta$ for all $\theta \in I_i$. For example, in Figure 5, we need to consider only $\tilde{\mathcal{N}}_0, \tilde{\mathcal{N}}_1, \tilde{\mathcal{N}}_3$, and $\tilde{\mathcal{N}}_6$.

From the definition of the time-expanded network, we can see that $v_{L_i - l_v}$ is contained in $\tilde{\mathcal{N}}_{L_i}$ for $i = 1, \dots, k$ and $v \in V_{\leq i}$. Recalling that $\tilde{b}(v_\theta) = g(v, \theta)$ for $v \in V$ and $\theta = 1, \dots, T$, $\tilde{b}_i = b_i$ holds. Thus, from \tilde{D}_{L_i} is isomorphic to $D[V_{\leq i}]$, the theorem follows. ■

Notice that Theorem 1 holds for only uniform path-lengths case. Here we show that Theorem 1 does not always hold for the general case. For a general dynamic network i.e., a dynamic network which does not satisfy the uniform path-lengths condition \mathcal{N} and a vertex v of \mathcal{N} , let l_v be the length of a shortest path from v to a sink s . Given a dynamic network \mathcal{N} illustrated in Figure 6, \mathcal{N} satisfies the condition of Theorem 1. However \mathcal{N} does not have a smooth evacuation. Figure 7 shows the time-expanded network $\mathcal{N}(7)$. The supply of v must wait at time step 0, or must be blocked w at time step 5.

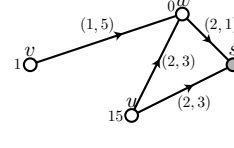


Fig. 6. Input dynamic network \mathcal{N} .

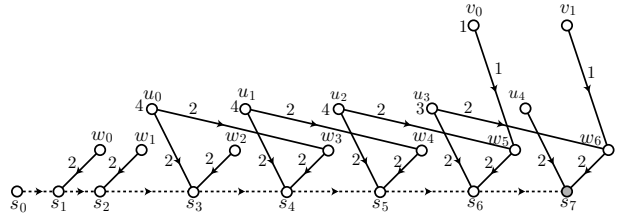


Fig. 7. Time-expanded network $\mathcal{N}(7)$.

If there exists no smooth evacuation in a given network, we must transform the network so as to have a smooth evacuation. One approach is to augment transit times of several arcs, i.e. to make a *buffer* by making supplies take a detour. For example, in the network of Figure 3, we can transform this network so as to have a smooth evacuation by augmenting the transit time of an arc wv by one (see Figure 8). It is clear that if we

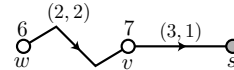


Fig. 8. Transformed network.

augment transit times of all arcs long enough, the network has a smooth evacuation. However, we want to minimize the total augmentation. This leads to the following optimization problem:

SMOOTH EVACUATION PROBLEM	
Input	Dynamic network \mathcal{N} .
Output	Minimum value of sum of augmentation such that \mathcal{N} has a smooth evacuation.

The existence of an efficient algorithm for this optimization problem including the uniform path-lengths case is open.

V. EVACUATION PATH PROBLEM

In the evacuation problem, we do not restrict paths from vertices to a sink. Thus, in an optimal solution, the path that supplies from a vertex to a sink s follow may not be unique.

However, it is desirable for the evacuation guidance that all the supplies at the same vertex flow through the same path to a sink. We call this problem the *evacuation path problem*, i.e., the evacuation problem under the constraint such that for each vertex $v \in V$, all the supplies of v take the same path from v to s . If only one vertex has a positive supply, this problem is called the *quickest path problem*, and has been widely studied. For the quickest path problem, $O(m^2 + mn \log n)$ time algorithms are independently presented by Chin et al. [4], Rosen et al. [15], and Martins et al. [14] where n and m denote $|V|$ and $|A|$, respectively. As for space complexity, the algorithm of [4] has to store the network whose size is equal to $O(m(n+m))$, where the two other algorithms require $O(n+m)$. Furthermore, Chen et al. [3] and Lee et al. [12] independently presented algorithms to determine the quickest path for any two vertices in $O(\log m)$ time, using $O(mn^2)$ preprocessing time. However, to the best of our knowledge, the evacuation path problem has not been studied.

A. Definition of Evacuation Path Problem

Here we formally define the *evacuation path problem* for a dynamic network $\mathcal{N} = (D = (V, A), c, \tau, b, s)$. Given a path p_v from v to s for each $v \in V - s$, we define a dynamic network flow $f_v: A \times \mathbb{Z}_+ \rightarrow \mathbb{R}_+$ in \mathcal{N} as follows. For $v \in V - s$, $e \in A$, and $\theta \in \mathbb{Z}_+$, we denote by $f_v(e, \theta)$ the amount of flow of supplies of v entering the tail of e at the time step θ . In this section, we call f *feasible* in \mathcal{N} if it satisfies the following four conditions, i.e., *path constraint*, *capacity constraint*, *flow conservation*, and *demand constraint*.

Path constraint: For any $v \in V - s$, $e \in A$ which p_v does not use, and $\theta \in \mathbb{Z}_+$,

$$f_v(e, \theta) = 0.$$

Capacity constraint: For any $e \in A$ and $\theta \in \mathbb{Z}_+$,

$$0 \leq \sum_{v \in V - s} f_v(e, \theta) \leq c(e).$$

Flow conservation: For any $v, w \in V - s$ and $\Theta \in \mathbb{Z}_+$,

$$\sum_{e \in \delta^+(v)} \sum_{\theta=0}^{\Theta} f_w(e, \theta) - \sum_{e \in \delta^-(v)} \sum_{\theta=0}^{\Theta - \tau(e)} f_w(e, \theta) \leq \begin{cases} b(v), & \text{if } v = w, \\ 0, & \text{if } v \neq w. \end{cases}$$

Demand constraint: There exists $\Theta \in \mathbb{Z}_+$ such that

$$\sum_{e \in \delta^-(s)} \sum_{\theta=0}^{\Theta - \tau(e)} \sum_{v \in V} f_v(e, \theta) = \sum_{v \in V} b(v). \quad (7)$$

Letting $P = \{p_v: v \in V - s\}$ and $\mathcal{F}(\mathcal{N}, P)$ be the set of all feasible flows, $\Theta(f)$ denotes the minimum time step Θ satisfying (7) for $f \in \mathcal{F}(\mathcal{N}, P)$. Let \mathcal{P} be the family of all sets of paths from each $v \in V$ to s . Given a dynamic network \mathcal{N} , the evacuation path problem $\text{EPP}(\mathcal{N})$ is formally defined as follows.

$$\text{EPP: minimize } \{\Theta(f): f \in \mathcal{F}(\mathcal{N}, P), P \in \mathcal{P}\}. \quad (8)$$

B. Intractability of the Evacuation Path Problem

First, we show that the decision version of $\text{EPP}(\mathcal{N})$ is \mathcal{NP} -hard where the decision version of $\text{EPP}(\mathcal{N})$ for $T \in \mathbb{Z}_+$ asks whether there exists $P \in \mathcal{P}$ such that there exists $f \in \mathcal{F}(\mathcal{N}, P)$ with $\Theta(f) \leq T$.

Theorem 2 The decision version of $\text{EPP}(\mathcal{N})$ is \mathcal{NP} -hard.

Proof: We prove the theorem by showing that any instance of an well-known \mathcal{NP} -complete problem PARTITION can be reduced to an instance of the decision version of $\text{EPP}(\mathcal{N})$.

PARTITION	
Instance	$K = \{a_1, a_2, \dots, a_p\} \subseteq \mathbb{Z}_+$
Question	Is there $K' \subseteq K$ such that $\sum_{a_i \in K'} a_i = \sum_{a_i \in K - K'} a_i$?

We construct the dynamic network $\mathcal{N} = (D = (V, A), c, \tau, b, s)$ to which an instance of PARTITION is reduced as follows. Let V be the union of $L_1 = \{v_1, \dots, v_p\}$ and $L_2 = \{w_1, w_2\}$ and $\{s\}$. Let A be the union of $\{v_i w_j: i = 1, \dots, p, j = 1, 2\}$ and $\{w_j s: j = 1, 2\}$ (see Figure 9). For every $e \in A$, we define $c(e) = 1$ and $\tau(e) = 0$. For $v_i \in L_1$, let $b(v_i) = a_i$, and let $b(w_1)$ and $b(w_2)$ be equal to zero.

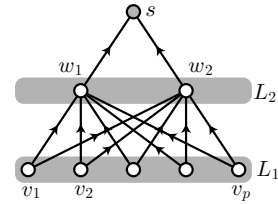


Fig. 9. Arcs between L_1 and L_2 (resp. L_2 and s) are directed to from L_1 to L_2 (resp. from L_2 to s).

For every $v \in L_1$, p_v has to use one of $w_1 s$ and $w_2 s$. Given $P \in \mathcal{P}$, let $V_1 \subseteq V$ (resp. $V_2 \subseteq V$) be the set of $v \in L_1$ such that p_v uses $w_1 s$ (resp. $w_2 s$). In this case, it is easy to see that $\Theta(f) = \max\{\sum_{v_i \in V_1} a_i, \sum_{v_i \in V_2} a_i\} - 1$ for any $f \in \mathcal{F}(\mathcal{N}, P)$. Thus, letting $T = (\sum_{a_i \in A} a_i)/2 - 1$, it is clear that $\Theta(f) \leq T$ if and only if the answer of PARTITION is “yes”. ■

From this theorem, we can see that $\text{EPP}(\mathcal{N})$ is \mathcal{NP} -hard. Thus, the next approach is to devise an approximation algorithm for this problem. How about *greedy algorithm*? That is, for each $v \in V - s$, we first solve the quickest path problem, i.e., $\text{EPP}(\mathcal{N})$ with $b(w) = 0$ for every $w \in V - v$. After this, we solve $\text{EPP}(\mathcal{N})$ such that we are given paths which are computed in the first step for all $v \in V - s$. However, the approximation ratio of this greedy algorithm is $\Omega(n)$ where n is the number of vertices in given network. Figure 10 shows the worst case instance. The vertex set of this worst case instance \mathcal{N} is the union of $\{s, w\}$ and $L = \{v_1, \dots, v_{n-2}\}$. The arc set is the union of $\{ws\}$ and $A_1 = \{v_i w: i = 1, \dots, n-2\}$ and $A_2 = \{v_i s: i = 1, \dots, n-2\}$ (see Figure 10). The supply of vertex in L is equal to one, and that of w is equal to zero. The capacity of every arc is equal to one. The transit time of every arc in $A_1 \cup \{ws\}$ is zero, and that of every arc in A_2 is one. The supply of every vertex in L is equal to one, and that of w is zero. In this instance, for each $v \in L$, the quickest path uses vw and ws . Thus, the value of the greedy algorithm is $n-2$. However, the optimal value of $\text{EPP}(\mathcal{N})$ is obtained when a path for only one vertex $v \in L$ uses vw and ws , and a path for every $w \in L - v$ uses ws . Hence the optimal value of $\text{EPP}(\mathcal{N})$ is equal to one. Thus, the approximation ratio of this greedy algorithm is $\Omega(n)$.

However, in the next subsection, we show that for the instance which have a grid structure and whose vertex size is small, the greedy algorithm gives a good solution.

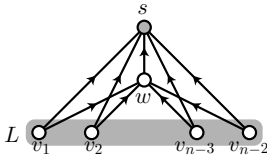


Fig. 10. Worst case instance.

C. Numerical example

In this subsection, we solve $EEP(\mathcal{N})$ by formulating this problem as a mixed integer programming. In this formulation, letting T be the upper bound of the optimal value, we use

$$\text{minimize } \sum_{e \in \delta^-(s)} \sum_{t \in \{0, \dots, T - \tau(e)\}} t \cdot f(e, t - \tau(e)) \quad (9)$$

as an objective function instead of (8) since these two objectives are equivalent [9]. (9) represents the minimization of the total time of evacuation.

We choose a 4×4 grid network as an input instance since an instance of a 5×5 grid network can not be solved in an hour. Moreover, we generate eight instances by changing capacities and transit times and supplies between one to ten.

We have done two experiments. In the first experiment, we solve a mixed integer programming, i.e., compute the optimal value. In the other experiment, we use the greedy algorithm described in the previous subsection. To solve a mixed integer programming, we used GLPK (GNU Linear Programming Kit) [2]. Furthermore, we wrote by C++ the code to find a quickest path for each vertex. These experiments were done on a PC with an Athlon 64 2.00 GHz with 960 MB memory. Table II shows the results of this experiment. In this table, “opt” represents the optimal value, and “time” represents the runtime in seconds spent to solve the problem. Moreover, “value” represents the objective value which the greedy algorithm found. Notice that for “Greedy”, although “time” does not contain the time spent to find a quickest path for all vertices, this time is no more than one second, and hence it is negligible.

TABLE II
RESULT OF EXPERIMENT.

No.	Optimal		Greedy	
	opt	time	value	time
1	878	567	888	3
2	1429	171	1513	6
3	635	29	635	1
4	1111	103	1130	3
5	1212	503	1271	5
6	1583	198	1694	2
7	1191	3104	1233	4
8	1723	3444	1862	6

As shown in Table II, in small size grid networks, the greedy algorithm can find a good solution. Moreover, the greedy algorithm can solve larger instance than those which we consider and this algorithm is very fast. Thus, in practical situation, the greedy algorithm is very useful.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented the results of the trade-off between computational complexity and the accuracy of model through

numerical examples arising from real data. Moreover, we studied the evacuation problem with several other features from the viewpoint of actual evacuation planning.

There is one more issue that we have not dealt with in this paper which is important from the viewpoint of actual evacuation planning. That is the difference of the transition speed depending on people. In the conventional model, each arc is given a single transit time. In the model taking the difference of the speed of people, there exist multicommodities in the network, and arcs have to be given several transit times for each commodity.

ACKNOWLEDGEMENT

This research is supported by the project *New Horizons in Computing*, Grant-in-Aid for Scientific Research on Priority Areas, MEXT Japan.

REFERENCES

- [1] <http://www.algorithmic-solutions.com/enledatech.htm>.
- [2] <http://www.gnu.org/software/glpk/>.
- [3] G.-H. Chen and Y.-C. Hung. On the quickest path problem. *Inf. Process. Lett.*, 46(3):125–128, 1993.
- [4] Y. L. Chen and Y. H. Chin. The quickest path problem. *Computers & OR*, 17(2):153–161, 1990.
- [5] L. Fleischer and M. Skutella. Quickest flows over time. *SIAM J. Computing*, 36(6), 2007.
- [6] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.
- [7] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. In *Proc. ICALP2003*, volume 2719 of *LNCS*, pages 397–409. Springer, 2003.
- [8] B. Hoppe and É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, February 2000.
- [9] J. J. Jarvis and R. D. Ratliff. Some equivalent objectives for dynamic network flow problems. *Management Science*, 28(1):106–109, 1982.
- [10] N. Kamiyama, N. Katoh, and A. Takizawa. An efficient algorithm for evacuation problems in dynamic network flows with uniform arc capacity. In *Proc. AAIM2006*, volume 4041 of *LNCS*, pages 231–242. Springer, June 2006.
- [11] N. Kamiyama, N. Katoh, and A. Takizawa. An efficient algorithm for the evacuation problem in a certain class of a network with uniform path-lengths. In *Proc. AAIM2007*, June 2007. (to appear).
- [12] D. T. Lee and E. Papadopoulou. The all-pairs quickest path problem. *Inf. Process. Lett.*, 45(5):261–267, 1993.
- [13] S. Mamada, T. Uno, K. Makino, and S. Fujishige. An $O(n \log^2 n)$ algorithm for the optimal sink location problem in dynamic tree networks. *Discrete Applied Mathematics*, 154(16):2387–2401, November 2006.
- [14] E. Q. V. Martins and J. L. E. Santos. An algorithm for the quickest path problem. *Operations Research Letters*, 20:195–198, 1997.
- [15] J. B. Rosen, S. Z. Sun, and G. L. Xue. Algorithms for the quickest path problem and the enumeration of quickest paths. *Computers & OR*, 18(6):579–584, 1991.