数理解析研究所講究録
1526 巻 2006 年 232-240

辺の容量が一定のネットワークにおける動的なフローを用いた避難計画問題に対する効率的なアルゴリズム

An Efficient Algorithm for the Evacuation Problem in Dynamic Network Flows with Uniform Arc Capacity and its Extension

神山 直之, 加藤 直樹, 瀧澤 重志
Naoyuki Kamiyama, Naoki Katoh, Atsushi Takizawa
京都大学大学院工学研究科建築学専攻
Department of Architecture and Architectural Engineering, Kyoto University

Abstract

In our previous paper [9], we proposed an $O(n \log n)$ time algorithm for the evacuation problem for a $\sqrt{n} \times \sqrt{n}$ grid network with uniform arc capacity. In this paper, we extend the classes of networks to which we can apply the algorithm of [9].

1 Introduction

Recently, 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. In our work, we adopt dynamic network flows[1] as a model for evacuation. A dynamic network flow is defined on a network which consists of a directed graph $D = (V, A)$ with capacity $c(e)$ and transit time $\tau(e)$ on every arc $e \in A$. For example, if we consider urban evacuation, vertices model buildings, rooms, exits and so on, and an arc models a pathway or a road connecting vertices. For an arc e, capacity $c(e)$ represents the number of people which can traverse the arc e per unit time, and transit time $\tau(e)$ denotes the time required to traverse e. Since Ford and Fulkerson [3], dynamic network flows have been studied extensively (see the survey by Kotnyek [10]).

Given a network with a single sink and initial supplies at vertices, the evacuation problem which we consider in this paper asks to find the minimum time horizon such that we can send all the initial supplies to a sink. This problem can be solved by the algorithm of Hoppe and Tardos [7] in polynomial time. However their running time is high-order polynomial, and hence 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 paper [9], we proposed an $O(n \log n)$ time algorithm for the evacuation problem for a $\sqrt{n} \times \sqrt{n}$ grid network with uniform arc capacity where n is number of vertices in given network. In this paper, we extend the classes of networks to which we can apply the algorithm of [9].

[1] A few authors (e.g., Fleischer in [2]) argue that the word "dynamic" is more consistently used for a problem with input that changes over time. Therefore these authors prefer to use the terms *flow over time* or *time dependent flow*.

1.1 Problem Formulation and notation

Let $\mathbb{R}_+$ and $\mathbb{Z}_+$ denote the set of nonnegative reals and nonnegative integers, respectively. We may represent a set $\{x\}$ of a single element by x. For any finite set X, we define $|X|$ as the number of elements belong to X.

We denote by $D = (V, A)$ a directed graph D which consists of a vertex set V and an arc set A. Moreover we denote by $e = (u, v)$ an arc e whose tail is u and head is v. In the case where an arc $e = (u, v)$ has no parallel arc, we may represent e by (u, v). A *path* $p = (e_1, e_2, \ldots, e_l)$ from a vertex $u \in V$ to a vertex $v \in V$ in D is a sequence of arcs belong to an arc set A which satisfies the following two conditions: (1) the head of e_i and the tail of e_{i+1} are the same vertex for any $i \in \{1, 2, \ldots, l-1\}$, (2) the tail of e_1 is u and the head of e_l is v. For any pair of subsets $X, Y \subseteq V$, we define $\delta(X, Y) = \{e = (x, y) : x \in X, y \in Y\}$, and we write $\delta^+(W)$ and $\delta^-(W)$ instead of $\delta(W, V - W)$ and $\delta(V - W, W)$, respectively. For any vertex $v \in V$, we define $P_v = \{w \in V : e = (w, v) \in A\}$. Moreover, for any pair of vertices $u, v \in V$, we denote by $\lambda(u, v)$ the local arc connectivity from u to v in D, i.e., the maximum number of the arc-disjoint paths from u to v in D.

Here we define a *dynamic network*. We denote by $\mathcal{N} = (D = (V, A), c, \tau, b, s)$ a dynamic network $\mathcal{N}$ which consists of the underlying directed graph $D = (V, A)$, a capacity function $c : A \to \mathbb{R}_+$ which represents the upper bound for the rate of flow that enters an arc per unit time, a transit time function $\tau : A \to \mathbb{Z}_+$ which represents the time required to traverse an arc, a supply function $b : V \to \mathbb{R}_+$ which represents the supply of each vertex, and a sink $s \in V$. Notice that for any arc e the transit time $\tau(e)$ is a nonnegative integer. Since we consider evacuation to a sink s, we assume that a sink s has no leaving arcs and no supply, and any vertex $v \in V$ is reachable to a sink s. For any vertex $v \in V$, we define $R_v = \{w \in P_s : w \text{ is reachable from } v \text{ in } D\}$. Finally we define a *length* of a path p in the underlying directed graph D of $\mathcal{N}$ as the sum of transit times of arcs on p, i.e., $\sum_{e \in p} \tau(e)$.

Here we define a *dynamic network flow* $f : A \times \mathbb{Z}_+ \to \mathbb{R}_+$ in a dynamic network $\mathcal{N} = (D = (V, A), c, \tau, b, s)$. For any arc $e \in A$ and time step $\theta \in \mathbb{Z}_+$, we denote by $f(e, \theta)$ the flow rate entering the arc e at the time step θ which arrives at the head of e at the time step $\theta + \tau(e)$. Notice that any time step is a nonnegative integer. We call f a *feasible dynamic network flow* in $\mathcal{N}$ if it satisfies the following three conditions, i.e., capacity constraint, flow conservation, and demand constraint [11].

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

$$0 \leq f(e, \theta) \leq c(e). \tag{1}$$

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

$$\sum_{e \in \delta^+(v)} \sum_{\theta=0}^{\Theta} f(e, \theta) - \sum_{e \in \delta^-(v)} \sum_{\theta=0}^{\Theta - \tau(e)} f(e, \theta) \leq b(v). \tag{2}$$

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

$$\sum_{e \in \delta^-(s)} \sum_{\theta=0}^{\Theta - \tau(e)} f(e, \theta) = \sum_{v \in V} b(v). \tag{3}$$

Here we give the intuitive understanding of the above three conditions. *Capacity constraint* ensures that the flow rate entering into any arc e at any time step θ is bounded by the capacity of e. *Flow conservation* ensures that the flow rate entering into any arc e at any time step θ is bounded by the sum of the flow rate arriving at the tail of e and the storage of the tail of e at the time step θ. *Demand constraint* ensures that all of supply flow into a sink s.

For a feasible dynamic network flow f in $\mathcal{N}$, let $\Theta(f)$ denote the completion time for f, i.e., the minimum time step Θ satisfying (3). The *evacuation problem* asks to find the minimum value of $\Theta(f)$ among all feasible dynamic network flows in $\mathcal{N}$. Given a dynamic network $\mathcal{N}$, the evacuation problem EP($\mathcal{N}$) is formally defined as follows:

$$\text{EP}(\mathcal{N})\text{: minimize}\{\Theta(f)\text{: } f \text{ is a feasible dynamic network flow in } \mathcal{N}\}.$$

Throughout this paper, n and m denote respectively the number of the vertices and the arcs in given network $\mathcal{N}$ for the evacuation problem EP($\mathcal{N}$).

1.2 Related works

Burkard, Dlaska, and Klinz [1] presented a strongly polynomial time algorithm for the evacuation problem in the case where only one vertex has a supply in given network. Unlike in the static network flow[2] problem, the evacuation problem can not be reduced to the case where only one vertex has a supply by using super-source which is connected with all vertex in given network. This is because the capacity of the arc connecting the super-source to a vertex v can not be set so that the total amount of flow that pass through this arc during the time horizon equal to the supply of v. The capacities of arcs in a dynamic network limit the flow rate at each time step. Hoppe and Tardos [7] gave the only known polynomial time algorithm for the evacuation problem. The algorithm of [7] solves the evacuation problem EP($\mathcal{N}$) by using $O(\mathcal{S}^2 \log^2(nCMT))$ minimum cost static network flow computations where $\mathcal{S}$, C, M, and $\mathcal{T}$ respectively denote the number of vertices with positive supplies, the maximum capacity of arcs, the sum of all supplies of vertices and the maximum transit time in $\mathcal{N}$. Their running time can be made strongly polynomial by using the parametric search technique of Megiddo [12].

As a special case, Hall, Hippler, and Skutella [6] consider the evacuation problem for a dynamic network such that for each vertex v a length of any path from v to a sink is the same value. Mamada, Uno, Makino, and Fujishige [11] consider the evacuation problem EP($\mathcal{N}$) for a dynamic network $\mathcal{N}$ with tree structure and presented an $O(n \log^2 n)$ time algorithm. For other special class, the algorithm of [9] solves the evacuation problem EP($\mathcal{N}$) for a dynamic network $\mathcal{N}$ with a $\sqrt{n} \times \sqrt{n}$ gird structure and uniform arc capacity in time $O(n \log n)$.

2 The Evacuation Problem for Grid Networks

First we define a *grid graph*. For simplicity, we assume a grid graph is on N^2 grid points $\{1, 2, \ldots, N\} \times \{1, 2, \ldots, N\}$ in the plane, and let $n = N^2$. Here a vertex is identified with

[2]In order to distinguish classical network flows from dynamic network flows, we call classic network flow *static network flow*.

(i,j) with $i \in \{1,2,\ldots,N\}$ and $j \in \{1,2,\ldots,N\}$. The distance between two vertices (i,j) and (i',j') is defined as $|i-i'|+|j-j'|$. Two vertices (i,j) and (i',j') are connected by an edge if and only if $|i-i'|+|j-j'|=1$ holds (Fig. 1(a)). The edge which connects v and v' is directed from v to v' if and only if the distance from v' to s is smaller than that from v to s (Fig. 1(b)). A dynamic network defined on a grid graph is called a *grid network*. We assume throughout this paper that, in dynamic networks we are concerned with, the capacities of all arcs take the same value $c \in \mathbb{R}_+$ and the transit times of all arcs take the same value $\tau \in \mathbb{Z}_+$. Notice that we define c and τ as not a function but an integer here. From this assumption, we use the notation $\mathcal{N}=(D=(V,A),b,s)$ for simplicity by omitting the capacity function and the transit time function. Moreover we assume a sink is an inner vertex, i.e. the in-degree of a sink is four (the other case can be similarly treated).

In a grid network $\mathcal{N}=(D=(V,A),b,s)$, for any vertex $v \in V$, we define l_v as the length of a path from v to a sink s. Notice that for any $v \in V$ l_v is unique in a grid network $\mathcal{N}$. Vertex set V is partitioned into layers according to the distance from s. Thus, a directed graph D can be viewed as a *layered graph*. A layered graph $D=(V,A)$ with a sink $s \in V$ is a directed graph consisting of several layers which partition V into subsets $V^0(=\{s\}), V^1, V^2, \ldots$ such that vertices $v \in V^i$ and $w \in V^j$ are connected by a directed arc $e=(v,w)$ only if $i-j=1$, and V^p denotes the set of all of vertices satisfying $l_v = p\tau$ (Fig. 1(c)). A dynamic network defined on a layered graph is called a *layered network*. Moreover we define $L\tau = \max\{l_v : b(v) > 0, v \in V\}$ for a grid network.

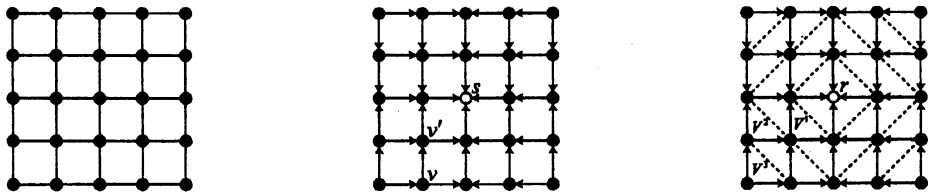

Figure 1: (a)Grid network (b)Direction of arcs (c)Layers of grid network

2.1 The Algorithm for Grid Networks

In this subsection, we consider the evacuation problem EP($\mathcal{N}$) for a grid network $\mathcal{N}=(D=(V,A),b,s)$. Our algorithm benefits from the structure of a grid graph. Let $P_s=\{u_1,u_2,u_3,u_4\}$. By the way of directing arcs of a grid graph, we can decompose V into eight subsets, U_1,U_2,U_3,U_4 and W_1,W_2,W_3,W_4 as in Fig. 2 where U_i denotes the set of vertices on horizontal or vertical axis whose supplies are all sent to a sink s through arc (u_i,s) and W_i denotes the set of vertices whose supplies are sent to a sink s through either (u_i,s) or (u_{i+1},s) (we assume throughout this section that the index i is given as $(i \bmod 4)+1$). Here let $H_i, i=1,2,3,4$ be a subgraph induced by $W_{i-1} \cup U_i \cup W_i \cup \{s\}$. For an optimal dynamic flow f for problem EP($\mathcal{N}$), it can be decomposed into four flows $f_i, i=1,2,3,4$ such that each f_i represents the flow of supplies which reaches s through arc (u_i,s). The proposed algorithm is based on the following four ingredients.

Theorem 2.1 There exists a subgraph H_i' of H_i which spans $W_{i-1} \cup U_i \cup W_i \cup \{s\}$ for $i = 1, 2, 3, 4$ such that H_i' are arc disjoint for $i \neq j$.

It is easy to see that the above theorem holds from Fig 3. Notice that that arc-disjoint subgraph H_i' are not uniquely determined.

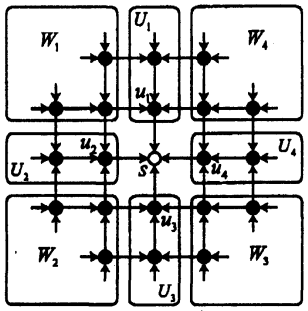

Figure 2: Decomposition of $\mathcal{N}$

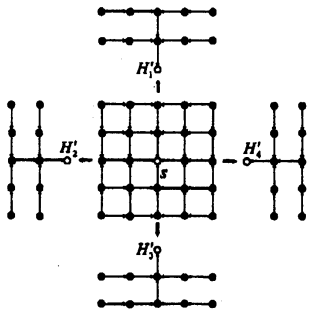

Figure 3: H_1', H_2', H_3', H_4'

Now suppose that for every $v \in W_i$ with $i = 1, 2, 3, 4$, the amounts of supply (denoted by $b_i(v)$ and $b_{i+1}(v)$ respectively) which reach r via arcs (u_i, s) and (u_{i+1}, s) respectively are fixed. Moreover we define $\mathcal{N}_i = (H_i, b_i, s)$ where for every $v \in U_i$ we define $b_i(v) = b(v)$.

Theorem 2.2 The optimal objective value for EP($\mathcal{N}_i$) for every $i \in \{1, 2, 3, 4\}$ does not depend on the choice of arc-disjoint subgraphs H_i', but remains the same.

Theorem 2.3 There exists an optimal dynamic flow f such that f_i and f_j does not share any arc for every $i \neq j$.

From these facts, when b_i and b_{i+1} are fixed for every $v \in W_i$ and every i with $i = 1, 2, 3, 4$, an optimal flow of EP($\mathcal{N}$) can be found by independently obtaining an optimal flow f_i^* for EP($\mathcal{N}_i$) for each $i \in \{1, 2, 3, 4\}$. Since the subgraph H_i' is a rooted tree, the solution of EP($\mathcal{N}_i$) can be given by simply specifying the supply at each $v \in W_{i-1} \cup U_i \cup W_i$. Thus, the problem EP($\mathcal{N}$) reduces to finding an optimal allocation of $b(v)$ to $b_i(v)$ and $b_{i+1}(v)$ for each $v \in W_i$ with $i = 1, 2, 3, 4$, and we call this problem *the optimal allocation problem for supplies*. Moreover, we prove the following theorem. Consequently, we can solve the evacuation problem for grid networks with uniform arc capacity efficiently as will be shown in Theorem 2.5.

Theorem 2.4 The optimal allocation problem for supplies can be transformed into the min-max resource allocation problem under network constraints [8, 5, 4].

The min-max resource allocation problem under network constraints is a kind of min-max flow problem with multiple sources and sinks in a static network [8, 5, 4] which is defined as follows. Suppose we are given a network with multiple sources and sinks such that a fixed amount of supply is associated with each source, and the cost function $\gamma_t(x_t)$ which is nondecreasing in x_t is associated with each sink t where x_t denotes the amount of flow entering t. Then the problem asks to find a (static) flow that minimizes the maximum of the cost functions of sinks.

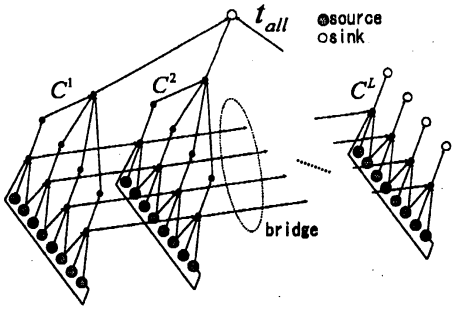

Figure 4: Illustration of the entire network constructed in Subsection 2.1

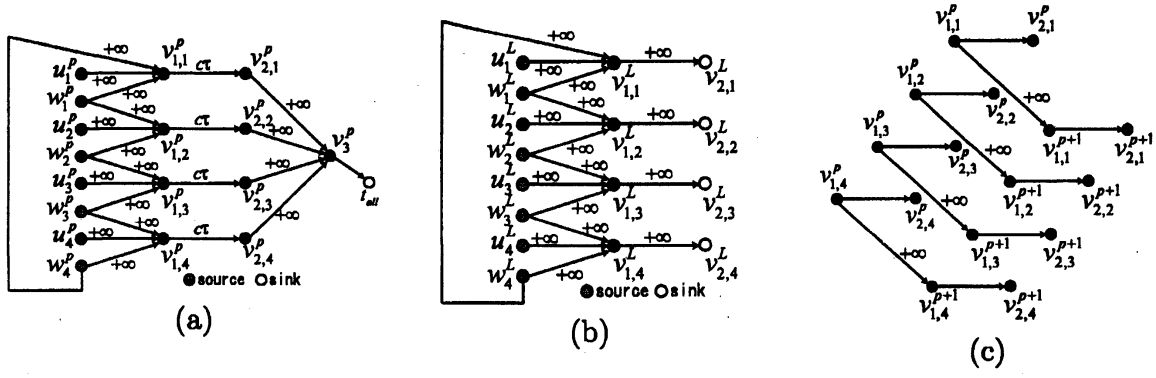

Figure 5: (a)p-th component C^p (b)L-th component C^m (c)p-th bridges

We will explain how we construct a (static) network (see Fig. 4) for which finding an optimal solution for the min-max resource allocation problem produces an optimal solution for the evacuation problem. The network to be constructed consists of L components $C^1, C^2, \ldots, C^L$. Each component C^p except C^L has four layers while C^L has three layers. The first layer of each component C^p has eight sources. The second and third layers consists of four vertices denoted by $v^p_{1,i}, v^p_{2,i}, i = 1,2,3,4$. The fourth layer consists of a single vertex v^p_3. The connection between the layers are as shown in Fig. 5(a). Only the arcs from the second to third layer have finite capacity $c\tau$ in C^p with $1 \leq p \leq L-1$ while the arcs in C^L have infinite capacity. The capacity of the other arcs is ∞. All vertices v^p_3 with $1 \leq p \leq L-1$ are connected to t_{all}.

The vertices $v^L_{2,i}, i = 1,2,3,4$ of C^L as well as t_{all} are sinks of this network which are associated with a cost function. The actual cost function for each $v^L_{2,i}, i = 1,2,3,4$ is equal to the amount the flow entering it. The cost function associated with t_{all} takes zero irrespective of the flow value entering it.

In addition to this, we prepare arcs between consecutive components. More precisely, as shown in Fig. 5(c), there is an arc from $v^p_{1,i}$ to $v^{p+1}_{1,i}$ for each p with $1 \leq p \leq L-1$ and i with $1 \leq i \leq 4$. The capacity of this arc is defined to be ∞. This arc is called a bridge.

It is known that the min-max resource allocation problem for the network with $|V|$ vertices, $|A|$ arcs and $|T|$ sinks can be solved in $O(|T|(|V||A|\log|V| + |T|\log\frac{M}{|T|}))$ time where M denotes the sum of supplies [8, 5, 4]. The second term in the parenthesis, i.e., $O(|T|\log\frac{M}{|T|})$, is the time required to solve the resource allocation problem without the

network constraints. Since our cost function associated with $v^L_{2,i}, i = 1, 2, 3, 4$ is linear, we can reduce the time to $O(1)$ (the details are omitted). In our case, $|T|$ is constant and $|V| = O(\sqrt{n}), |A| = O(\sqrt{n})$, thus the running time becomes $O(n \log n)$. This proves the following theorem.

Theorem 2.5 The evacuation problem for a grid network with uniform arc capacity can be solved in $O(n \log n)$ time.

3 Extension of the Algorithm of [9]

It is easy to see that the algorithm of [9] can be extended to a general layered network $\mathcal{N}$ such that (1) the length of any path from a vertex v to a sink s take the same value, and (2) the underlying layered graph $D = (V, A)$ (we allow D to have multiple arcs) contains arc-disjoint layered subgraphs $H_1, H_2, \ldots, H_k$ which spans $U_1, U_2, \ldots, U_k$ and includes $e_1, e_2, \ldots, e_k$ respectively, where we define $\delta^-(s) = \{e_1, e_2, \ldots, e_k\}$ and U_i is the set of vertices from which the tail of e_i is reachable in D. Thus, the result can also be generalized to the case where the arc capacity is a multiple of c by regarding the arc as multiple ones as long as the resulting layered graph satisfies the requirement just mentioned above. In this section we consider the conditions under which layered graphs contain such arc-disjoint layered subgraphs.

Here we consider a layered graph $D = (V, A)$ with a sink $s \in V$. We denote a layer whose distance from s is i as V^i. Notice that $V^0 = \{s\}$. We define $\delta^-(s) = \{e_1, e_2, \ldots, e_k\}$ and U_i be the set of vertices form which the tail of e_i is reachable in D. We prove the following lemma.

Lemma 3.1 Given a layered network $D = (V, A)$ with a sink $s \in V$, there exist k aborescences $T_i = (U_i, A_i)$ for each $i \in \{1, 2, \ldots, k\}$ such that every T_i is rooted at s, $e_i \in A_i$, $A_i \subseteq A$ and $A_i \cap A_j = \emptyset$ $(i \neq j)$ hold if and only if for any $v \in V$ we have $\lambda(v, s) = |\delta(R_v, s)|$.

Proof: If there exist k aborescences satisfying the lemma statement, it is easy to see that for any $v \in V$ we have $\lambda(v, s) = |\delta(R_v, s)|$. We then prove the "if-part".

Assume that for any $v \in V$ we have $\lambda(v, s) = |\delta(R_v, s)|$. We prove there exist k aborescences satisfying the lemma statement by induction on the number of layers i.

In the case for $i = 1$, it is easy to see that the lemma holds.

Next we assume for $i = j$ the lemma holds. Notice that since D is a layered graph the parents of a vertex of V^{j+1} belong to V^j from the definition of a layered graph. Since $\lambda(v, s) = |\delta(R_v, s)|$ holds, the number of the arcs whose tail is v is at least $|\delta(R_v, s)|$. We have

$$R_v = \bigcup_{w:\, e=(v,w)\in A} R_w.$$

For convenience, let the arcs whose tail is v be $\hat{e}_1, \hat{e}_2, \ldots$ (Figure 6).

Here we define the bipartite graph $G = ((V^+, V^-), E)$ as follows. An element of V^+ corresponds to an element of $R(v)$ and an element of V^- corresponds to an element in the set of arcs whose tail is v. Then, a vertex $v^+ \in V^+$ and a vertex $v^- \in V^-$ are joined

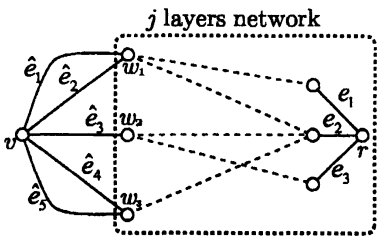

Figure 6: $j+1$-th layer

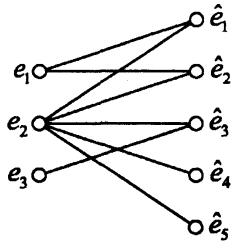

Figure 7: Bipartite graph of v in Fig 6

by an edge in E if and only if the head of the arc which corresponds to v^- is reachable to the tail of the arc which corresponds to v^+ (Figure 7). By induction hypothesis, there exists a matching which saturates V^+ in the bipartite graph $G = ((V^+, V^-), E)$ defined above.

We then prove that there always exists a matching as described above. Here we use Hall's theorem.

Theorem 3.1 ([13]) A bipartite graph $G = ((V^+, V^-), E)$ has a matching which saturates all nodes of V^+ if and only if for any $H \subseteq V^+$, $|H| \leq |N(H)|$ holds where $N(H)$ is the set of vertices adjacent to some element of H.

We prove the existence of such matching as described above by contradiction. Assume that there exists $H \subseteq V^+$ with $|H| > |N(H)|$. This contradicts the fact that there are at least $|\delta(R_v, s)|$ arc-disjoint paths. This completes the proof. □

From Lemma 3.1, we can easily prove the following theorem.

Theorem 3.2 The evacuation problem for a layered network $\mathcal{N} = (D = (V, A), b, s)$ with uniform arc capacity which satisfies $\lambda(v, s) = |\delta(R_v, s)|$ for any $v \in V$ can be solved in $O(m + k^3 n^2 \log n)$ time where we define $k = |P_s|$.

Proof: To finish the proof of the theorem, we have to prove the correctness of the time complexity. The first term, i.e., $O(m)$ is the time required to obtain R_v for any $v \in V$ by depth-first search. The second term is the time required to solve the min-max resource allocation problem produces an optimal solution for the evacuation problem. Since the time complexity depends on the size of the network for which finding an optimal solution for the min-max resource allocation problem produces an optimal solution for the evacuation problem. Note that the example of the network for the evacuation problem for grid networks is shown as in Figure 4. Though we omit the details, the number of vertices is $O(kn)$, the number of arcs is $O(kn)$, and the number of sinks is $O(k)$. Thus, the running time of the algorithm is $O(k^3 n^2 \log n)$ by Theorem 2.5. □

Acknowledgements

This research is supported by JSPS Grant-in-Aid for Scientific Research on priority areas of New Horizons in Computing.

References

[1] R.E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR-Methods and Models of Operations Research*, 37:31–58, 1993.

[2] L. Fleischer. Faster algorithm for the quickest transshipment problem. *SIAM J. on Optimization*, 12(1):18–35, 2001.

[3] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[4] S. Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, 5(2):186–196, May 1980.

[5] S. Fujishige. Nonlinear optimization with submodular constraints. In *Submodular Functions and Optimization*, volume 58, pages 223–250. Elsevier Science, North-Holland, 2nd edition, 2005.

[6] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. In *Automata, Languages and Programming, 30th International Colloquium (ICALP 2003)*, volume 2719 of *LNCS*, pages 397–409. Springer, 2003.

[7] B. Hoppe and É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, February 2000.

[8] T. Ibaraki and N. Katoh. Resource allocation problems under submodular constraints. In *Resource Allocation Problems : Algorithmic Approaches*, pages 144–176. MIT Press, Cambridge, MA, 1988.

[9] N. Kamiyama, N. Katoh, and A. Takizawa. An efficient algorithm for evacuation problems in dynamic network flows with uniform arc capacity. *IEICE Transaction on Fundamentals*, E89-D(8):2372–2379, August 2006.

[10] B. Kotnyek. An annotated overview of dynamic network flows. Technical Report RR-4936, Inria Sophia Antipolis, September 2003.

[11] S. Mamada, T. Uno, K. Makino, and S. Fujishige. An $O(n \log^2 n)$ algorithm for a sink location problem in dynamic tree networks. *Discrete Applied Mathematics*, to appear.

[12] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operation Research*, 4:414–424, 1979.

[13] W.R. Pulleybland. Matchings and extensions. In R.L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume 1, pages 179–233. MIT Press, 1995.