A Decentralized Algorithm for Network Flow Optimization in Mesh Networks

Kiyoshi Nakayama*, Toshio Koide†
*Dept. of Computer Science, University of California, Irvine, USA
†Knowledge Discovery Research Laboratories, NEC Corporation, Japan
Email: {kiyoshi.nakayama, toshio-k}@ieee.org

***Abstract*—In order to evaluate total throughput against given traffics in an entire network, we formulate a minimum cost flow problem with quadratic edge functions, which we call *Network Flow Optimization (NFO)* problem in this paper. The problem with quadratic flow costs has been proved to be NP-complete. However, by dividing a network into a set of loops that represents a linear vector space, the problem can efficiently be solved. The theory that deals with the nature of loops of a graph is called tie-set graph theory where a *tie-set* represents a set of edges that constitute a loop. The theory of tie-sets has played a significant role in solving core problems in the domain of circuits and power systems as in applications of Kirchhoff's theory. Therefore, we propose a novel decentralized algorithm based on tie-set graph theory to optimize network flows in a mesh network. Global optimization can be achieved by iterative distributed computation where flows within a loop are locally optimized. Simulation results demonstrate the optimal allocation of network flows and show the superiority over the multi-path routing method.**

I. Introduction

Network flow problems have been studied for a long time since the birth of graph and network theory [1], [2] as in the maximum flow problem with given capacities of a network [3]. In the field of circuits and power systems, the Kirchhoff's Current and Voltage Laws also have a very long history and have demonstrated the effectiveness in controlling power flows and designing fault-tolerant reliable systems even in a complicated non-planar meshed topology [4]. While the concept of Kirchhoff's Theory seems quite useful for various applications in network systems, the nature of distributed computations makes the integration of those theories difficult. Bridging the gap between the theoretical basis and complexity in decentralizing computations must lead to breakthrough in performance improvement of network systems' applications giving an enough motivation to overcome this challenge. Therefore, we develop a decentralized algorithm to be able to make the Kirchhoff's theory function in the network systems domain.

Multi-path (MPATH) routing techniques have been proposed in the recent literature [5], [6], [7] dealing with creating efficient loop-free multi-paths. MPATH routing techniques are often applied to traffic engineering with Multi-Protocol Label Switching (MPLS) [8], [9], [10]. While these schemes with distributed computing have been used to achieve an efficient traffic allocation to improve total throughput in a network, distributing network flows to several paths has not become a radical solution to optimize network flows. This is because the evaluation of total throughput is modeled with a network flow problem with quadratic edge functions that has been proved to be NP-complete [11] and difficult to solve in a decentralized fashion. Therefore, it has been difficult to radically solve the problem in a distributed manner even if many efforts have been made to improve throughput with reroutings of network flows. This gives rise to the study of network flow optimization integrating the notion of tie-sets in Kirchhoff's Theory that has played a significant role in solving convex optimization problems with quadratic edge functions.

A network flow optimization problem in this paper is to balance the load factor on every edge by redirecting flows from heavily loaded links to lightly loaded links on the basis of a μ-dimensional linear vector space[1] defined with the tie-sets, which consists of local optimization units representing loops in a network. The previous work on network flow optimization provides a theoretical basis that proves that a flow problem with convex edge functions can be formulated with a set of μ tie-sets where a globally optimum point can be reached if the necessary and sufficient condition is given and solved on tie-set basis [12], [13].

In this paper, we propose a completely autonomous control method and a decentralized algorithm to solve the network flow optimization problem with simulation experiments using ARPANET network. The comparison experiment against the MPATH flow distribution technique also shows the superiority of the proposed method with a modest delay in convergence.

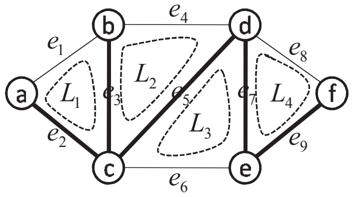
II. Tie-Set Graph Theory

As the tie-set graph theory is described in [12], [13], [14] in detail, we provide the basis for the unfamiliar reader.

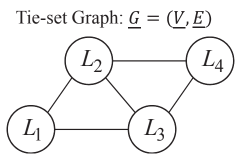
A. Fundamental System of Tie-sets

For a given graph $G = (V, E)$ with a set of vertices $V = \{v_1, v_2, ..., v_n\}$ and a set of edges $E = \{e_1, e_2, ..., e_m\}$, let $L_i = \{e^i_1, e^i_2, ...\}$ be a set of all the edges that constitutes a loop in G. The set of edges L_i is called a *tie-set* [4]. Let T and $\overline{T}$ respectively be a spanning tree and a cotree of G, where $\overline{T} = E - T$. $\mu = \mu(G) = |\overline{T}|$ is called the *nullity* of a graph. For $l \in \overline{T}$, $T \cup \{l\}$ includes one tie-set. Focusing on a subgraph $G_T = (V, T)$ of G and an edge $l = (a, b) \in \overline{T}$, there exists only one elementary path $P_T(b, a) \subseteq T$ whose origin

[1] μ is the nullity of a graph as defined in Section II-A



(a) A fundamental system of tie-sets $\{L_1, L_2, L_3, L_4\}$.



Tie-set Graph: $\underline{G} = (\underline{V}, \underline{E})$
L_1
L_2
L_3
L_4

(b) The tie-set graph of Fig. 1(a).

Fig. 1. A fundamental system of tie-sets and its tie-set graph [12]. Thick and thin lines are links of a tree T and a cotree $\overline{T}$, respectively.

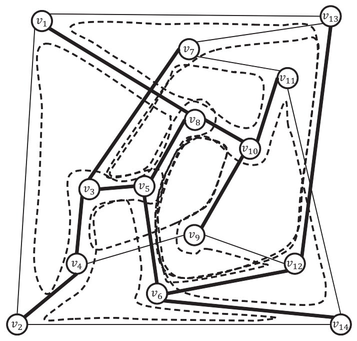


Fig. 2. An example of a fundamental system of tie-sets in a non-planar graph.

is b and terminal is a in G_T. Then, a *fundamental tie-set* that consists of the path P_T and the edge l is uniquely determined as $L(l) = \{l\} \cup P_T(b,a)$. There are μ fundamental tie-sets in G and they are called a *fundamental system of tie-sets*. If a graph G is bi-connected, a fundamental system of tie-sets $\mathbb{L}_B = \{L_1, L_2, ...L_\mu\}$ covers all the vertices and edges as shown in the planar graph of Fig. 1(a) and the non-planar graph of Fig. 2.

Fundamental tie-sets are independent of each other; any fundamental tie-set cannot be obtained by the calculus $\oplus$[2] among other tie-sets.

B. Tie-set Graph

A graph $\underline{G} = (\underline{V}, \underline{E})$ is defined as a *tie-set graph*, where a set of vertices $\underline{V}$ corresponds to a fundamental system of tie-sets $\{L_1, L_2, ..., L_\mu\}$, and a set of edges $\underline{E}$ is denoted as $\{\underline{e}(L_i, L_j)\}, (i \neq j)$, which represent the connections among tie-sets. In this paper, $\underline{e}(L_i, L_j)$ is determined by the set of common vertices of L_i and L_j. Let $V(L_i)$ be a set of all the vertices included in a tie-set L_i. If $V(L_i) \cap V(L_j) \neq \emptyset$, L_i and L_j have an edge $\underline{e}(L_i, L_j)$. Each fundamental tie-set of a given graph G is uniquely mapped to the specific tie-set graph $\underline{G}$ as shown in Fig. 1(b).

III. Network Flow Optimization Problem

In this section, we formulate a Network Flow Optimization problem.

[2]The definition of $\oplus$ for a set A and a set B is defined as follows: $A \oplus B = (A - B) \cup (B - A) = (A \cup B) - (A \cap B)$.

An information flow network $N = (G, F, C, s, t)$ is given with a directed graph $G = (V, E)$, $(|V| = n, |E| = m)$, a set of edge flows $F = \{f_k\}, (k = 1, 2, \ldots, m)$, a set of link capacities (maximum flows) $C = \{c_k\}, (k = 1, 2, \ldots, m)$, a source node s, and a sink node t. A link e_k corresponds to a k-th communication link (channel) in a network. An edge flow f_k is bounded by

$$-c_k \leq f_k \leq c_k, \quad \text{for } e_k \in E. \tag{1}$$

Each link is directed with an arbitrarily defined direction, and a link from node u to v is denoted interchangeably by either $e(u,v) \in E$ or $u \to v$. Let $u : u \to v$ and $w : v \to w$ denote the set of predecessors and successors of node v in the directed graph, respectively. An edge flow $f(u,v)$ is passing traffic quantity over a link $e(u,v)$ from vertex u to v. When a flow $f(u,v)$ passes along the direction of an edge $e(u,v)$ then $f(u,v) > 0$; otherwise $f(u,v) < 0$. As an edge flow can either be positive or negative, the following holds true at node v:

$$\sum_{w:v\to w} f(v,w) - \sum_{u:u\to v} f(u,v) = \begin{cases} 0, & v \neq s,t \\ \mathcal{F}, & v = s \\ -\mathcal{F}, & v = t \end{cases} \tag{2}$$

A load factor $g(f_k)$ on a link e_k is defined as

$$g(f_k) = \frac{f_k}{c_k}, \quad \text{for } e_k \in E. \tag{3}$$

To quantitatively assess the load factor, let us define a quadratic *edge function* $\psi(f_k)$ where

$$\psi(f_k) = g^2(f_k), \quad \text{for } e_k \in E. \tag{4}$$

Then, we define a *network flow function* Φ_N of the network N as

$$\Phi_N = \sum_{e_k \in E} \psi(f_k), \tag{5}$$

and we want to minimize Φ_N to balance the load factors.

The network flow optimization problem can be considered as an optimization problem over an *edge flow vector* $\mathbf{F} = [f_1, f_2, ..., f_k]$. A diagonal matrix $\mathbf{D}$ whose elements are capacities c_k is denoted as $\mathbf{D} = \text{diag}(c_1, c_2, ..., c_k)$. Then, the network flow function Φ_N is expressed by using $\mathbf{F}$ as

$$\Phi_N = \mathbf{F}\mathbf{D}^{-2}\mathbf{F}^{\mathrm{T}}. \tag{6}$$

Our objective is to find the solution $\mathbf{F}$ that minimizes the network flow function Φ_N in (6). Now we show how to convert the problem in (6) to an optimization problem over tie-set flows.

A flow x_λ circulating along a tie-set $L_\lambda \in \mathbb{L}_B$ is defined as a *tie-set flow* with respect to a tie-set L_λ. A tie-set flow x_λ has its direction as shown in Fig. 3, where $x_\lambda > 0$ if it flows in the same direction as the direction defined for L_λ, and $x_\lambda < 0$ otherwise. Then, a *tie-set flow vector* $\mathbf{x}$ with respect to a fundamental system of tie-sets $\mathbb{L}_B$ is defined as

$$\mathbf{x} = [x_1, x_2, ..., x_\mu]. \tag{7}$$

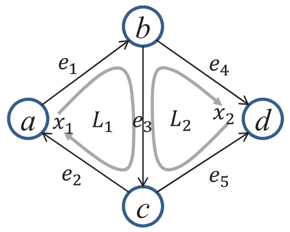


Fig. 3. An example of the direction of edges and tie-set flows.

The set of all possible tie-set flow vectors is defined as a *tie-set flow vector space*. The tie-set flow vector space forms Euclidean vector space $\mathbb{R}^{\mu}$ with dimension μ.

Let $\mathbf{B} = [b_{\lambda k}] \in \mathbb{R}^{\mu \times m}$ be a *tie-set matrix* of a graph G with respect to $\mathbb{L}_B$, where

$$b_{\lambda k} = \begin{cases} 0 & e_k \notin L_\lambda \\ 1 & e_k \in L_\lambda,\ \text{the same direction as } x_\lambda \\ -1 & e_k \in L_\lambda,\ \text{the opposite direction to } x_\lambda \end{cases} \tag{8}$$

For example, $\mathbf{B} = [b_{\lambda k}]$ in Fig. 3 is determined as follows:

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 \end{bmatrix}$$

We select an arbitrary set of *initial flows* $\{\xi_k\}$, and define an *initial flow vector* as $\Xi = [\xi_1, ..., \xi_m]$. A set of initial flows satisfies (1) and (2). Then, the edge flow vector $\mathbf{F}$ is expressed as

$$\mathbf{F} = \mathbf{x}\mathbf{B} + \Xi. \tag{9}$$

Using (9), the network flow function (6) is transformed into a function of the tie-set flow vector $\mathbf{x}$ as

$$\begin{aligned} \Phi_N(\mathbf{x}) &= (\mathbf{x}\mathbf{B} + \Xi)\mathbf{D}^{-2}(\mathbf{x}\mathbf{B} + \Xi)^{\mathrm{T}} \\ &= \mathbf{x}\mathbf{B}\mathbf{D}^{-2}\mathbf{B}^{\mathrm{T}}\mathbf{x}^{\mathrm{T}} + 2\mathbf{x}\mathbf{B}\mathbf{D}^{-2}\Xi^{\mathrm{T}} + \Xi\mathbf{D}^{-2}\Xi^{\mathrm{T}}, \end{aligned} \tag{10}$$

since $\mathbf{x}\mathbf{B}\mathbf{D}^{-2}\Xi^{\mathrm{T}} = \Xi\mathbf{D}^{-2}\mathbf{B}^{\mathrm{T}}\mathbf{x}^{\mathrm{T}}$. Therefore, the network flow function (10) is expressed as

$$\Phi_N(\mathbf{x}) = \mathbf{x}\mathbf{M}\mathbf{x}^{\mathrm{T}} + \mathbf{x}\mathbf{N} + \mathbf{e}, \tag{11}$$

where $\mathbf{M} := \mathbf{B}\mathbf{D}^{-2}\mathbf{B}^{\mathrm{T}}$, $\mathbf{N} := 2\mathbf{B}\mathbf{D}^{-2}\Xi^{\mathrm{T}}$, and $\mathbf{e} := \Xi\mathbf{D}^{-2}\Xi^{\mathrm{T}}$.

Let $\mathbf{C} = [c_1, c_2, ..., c_k]$ be a *capacity vector* consisting of the capacities c_k. Given $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, and Ξ, the *Network Flow Optimization (NFO)* problem is

$$\min \quad \Phi_N(\mathbf{x}) = \mathbf{x}\mathbf{M}\mathbf{x}^{\mathrm{T}} + \mathbf{x}\mathbf{N} + \mathbf{e}, \tag{12}$$

$$\text{s. t.} \quad -\mathbf{C} \leq \mathbf{x}\mathbf{B} + \Xi \leq \mathbf{C}. \tag{13}$$

For a given initial flow vector Ξ, the NFO problem is to find μ tie-set flows such that the value of the network flow function is minimum, i.e. find the point $\mathbf{x} = \mathbf{x}^* \in \mathbb{R}^{\mu}$ at which $\Phi_N(\mathbf{x}^*)$ is minimum in the tie-set flow vector space $\mathbb{R}^{\mu}$. As the network flow function $\Phi_N(\mathbf{x})$ is differentiable on $\mathbb{R}^{\mu}$, the necessary and sufficient condition for $\mathbf{x} = \mathbf{x}^*$ to be the optimal point of NFO problem is that the gradient of $\Phi_N(\mathbf{x}^*)$ is zero [13]. Namely, at $\mathbf{x} = \mathbf{x}^*$,

$$\nabla\Phi_N(\mathbf{x}^*) = \left(\frac{\partial \Phi_N(\mathbf{x}^*)}{\partial x_1}, ..., \frac{\partial \Phi_N(\mathbf{x}^*)}{\partial x_\mu} \right) = \mathbf{0}. \tag{14}$$

TABLE I

NOTATIONS AND DEFINITIONS FOR ADCT

MV $y_\lambda(t)$	*Measurement Vector*. MV $y_\lambda(t)$ contains various information of nodes $v_i \in V(L_\lambda)$ and links $e_k \in L_\lambda$ at time t such as current edge flows.
TA	*Tie-set Agent*. An autonomous agent that constantly navigates a tie-set to bring the current MV $y_\lambda(t)$ with state information of L_λ to its leader node.
TEF $\Phi(L_\lambda)$	*Tie-set Evaluation Function*. A function that evaluates a tie-set based upon the current MV $y_\lambda(t)$ with certain predefined criterion.
TEFM	*Tie-set Evaluation Function Message*. A message used to exchange the value of TEF $\Phi(L_\lambda)$ with adjacent tie-sets $\mathbb{L}^a_\lambda$.
TF $\zeta(L_\lambda)$	*Tie-set Flag*. When $\zeta(L_\lambda) = 0$, a tie-set L_λ is stand-by; otherwise L_λ is in process ($\zeta(L_\lambda) = 1$).
TFS	*Tie-set Flag Signal*. A signal to notify the state of TF $\zeta(L_\lambda)$.

IV. Distributed Algorithms for Network Flow Optimization Problem

Now we show how to solve NFO problem by local optimization with respect to μ independent tie-sets that leads to the global optimization.

A. Fundamental Distributed Algorithms based on Tie-set

1) State Information of a Node: Each node v_i mainly has the following state information [15]:

Tie-set Information: Information of fundamental tie-sets to which v_i belongs. When $v_i \in V(L_\lambda)$, it is defined that v_i belongs to L_λ and has information of L_λ.

The node c in Fig. 1(a), for example, has state information of $\{L_1, L_2, L_3\}$ as Tie-set Information. Each node executes a Distributed Algorithm for Tie-set Information Configuration (DATIC) to recognize fundamental tie-sets to which the node belongs [16].

2) Communications among Tie-Sets (CAT): In addition to state information described in IV-A1, a leader node v^λ_l of a tie-set L_λ has the additional information below to conduct CAT:

Adjacent Tie-sets $\mathbb{L}^a_\lambda = \{L^\lambda_1, L^\lambda_2, ...\}$: An adjacent tie-set L_j of L_λ is determined according to $\underline{e}(L_\lambda, L_j) \in \underline{E}$ of $\underline{G}$.

For instance, adjacent tie-sets of L_1 in Fig. 1(a) are $\mathbb{L}^a_1 = \{L_2, L_3\}$ so that L_1 constantly communicates with $\{L_2, L_3\}$.

B. Autonomous Distributed Control in Tie-set

We propose an *Autonomous Distributed Control in Tie-set (ADCT)* that is conducted in a leader node in each tie-set. The notations and definitions for ADCT are found in TABLE I. In this research, we use $\Phi(L_\lambda) = \left|\frac{d}{dx_\lambda}\phi(x_\lambda)\right|$ as TEF where

$\phi(x_\lambda)$ is defined in (15) in Section IV-C. Then, the procedure of ADCT is explained as follows:

1) Initialization: In **Initialization**, TEF of a tie-set L_λ is set as $\Phi(L_\lambda) = 0$. TF of L_λ is also set as $\zeta(L_\lambda) = 0$. Then, L_λ calls **Send**.

2) Send: In **Send**, the value of TEF $\Phi(L_\lambda)$ is calculated based upon the current MV $y_\lambda(t)$ provided by TA that constantly navigates L_λ. After calculating $\Phi(L_\lambda)$, L_λ writes its TEF value into TEFM. Then, L_λ sends TEFM to all the adjacent tie-sets $\mathbb{L}^a_\lambda$.

3) Receive: **Receive** is called when L_λ receives TEFM. Until L_λ receives TEFM from all the adjacent tie-sets $\mathbb{L}^a_\lambda$, L_λ stands by to receive another TEFM. After receiving TEFM form all of $\mathbb{L}^a_\lambda$, L_λ calls **Compare**.

4) Compare: In **Compare**, L_λ compares its value of $\Phi(L_\lambda)$ with those of adjacent tie-sets. If the value of $\Phi(L_\lambda)$ is the largest among those of all the adjacent tie-sets, L_λ sets its TF as $\zeta(L_\lambda) = 1$, otherwise $\zeta(L_\lambda) = 0$. If the value of $\Phi(L_\lambda)$ is the same as $\Phi(L_j)$, L_λ uses another TEF $\Phi(L_\lambda)$ to decide the process priority such as $\Phi(L_\lambda) = \text{Random}$. Then, L_λ calls **Optimize**.

5) Optimize: If $\zeta(L_\lambda) = 1$, L_λ conducts Decentralized Algorithm for Tie-set Flow Optimization (DATFO) described in Algorithm 1. Then, L_λ sets its TF as $\zeta(L_\lambda) = 0$, and calls **Notify**.

6) Notify: In **Notify**, L_λ sends TFS to each adjacent tie-set $L_j \in \mathbb{L}^a_\lambda$ to notify that $\zeta(L_\lambda) = 0$.

7) Confirm: **Confirm** is called when L_λ receives TFS from all the adjacent tie-sets $\mathbb{L}^a_\lambda$ after **Optimize**. In case that **Optimize** is not finished, TFS is temporarily stored in L_λ. Until L_λ receives TFS from all of $\mathbb{L}^a_\lambda$, L_λ waits for another TFS. After receiving TFS form all of $\mathbb{L}^a_\lambda$, L_λ confirms that each TF of $L_j \in \mathbb{L}^a_\lambda$ is $\zeta(L_j) = 0$. Then, L_λ calls **Send** again so that ADCT is iterated.

The flowchart of ADCT is described in Fig. 4.

C. Decentralized Algorithm for Tie-set Flow Optimization

Decentralized Algorithm for Tie-set Flow Optimization (DATFO) is described in Algorithm 1 and called by ADCT.

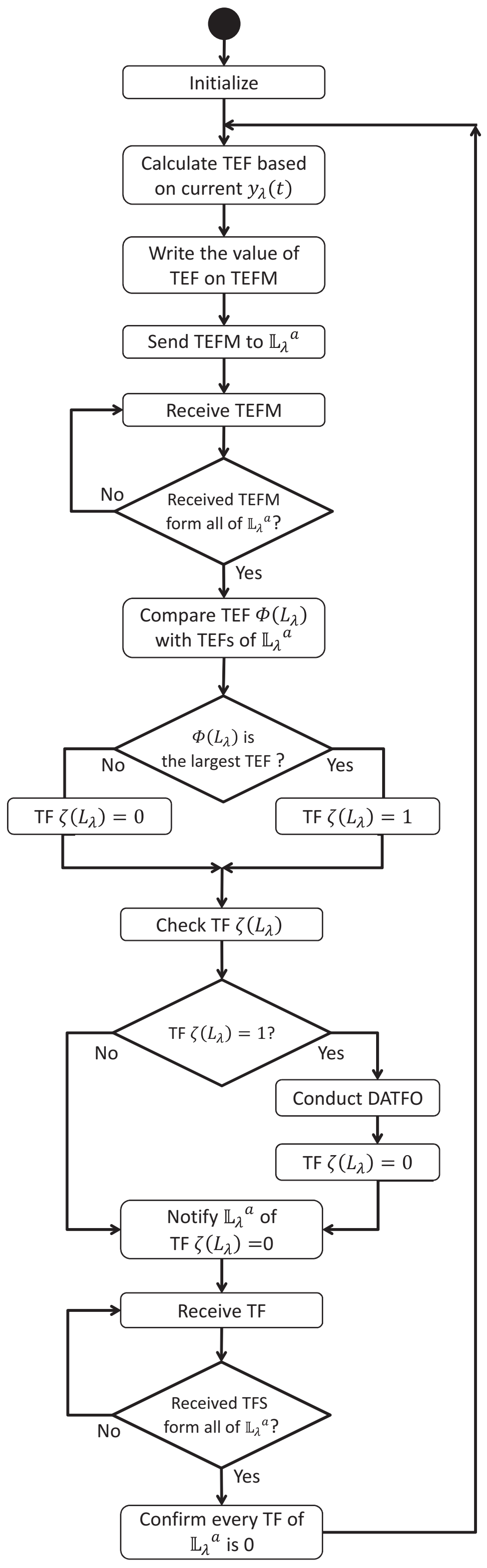
In STEP 0, DATFO initializes the value of a current tie-set flow x_λ of L_λ as 0. The information of current edge flows $\Xi^\lambda = \{\xi_k\}$ of $e_k \in L_\lambda$ has been provided by MV $y_\lambda(t)$ included in TA at time t.

Let $\phi(x_\lambda)$ be a *Tie-set Flow Optimization (TFO)* function of a tie-set L_λ defined as

$$\phi(x_\lambda) = \sum_{e_k \in L_\lambda} \left(\frac{b_{\lambda k} x_\lambda + \xi_k}{c_k} \right)^2. \tag{15}$$

In STEP 1, DATFO calculates the optimal tie-set flow x_λ with the current edge flows $\Xi^\lambda = \{\xi_k\}$ to satisfy

$$\begin{aligned} \frac{d}{dx_\lambda}\phi(x_\lambda) &= \frac{d}{dx_\lambda} \sum_{e_k \in L_\lambda} \left(\frac{b_{\lambda k} x_\lambda + \xi_k}{c_k} \right)^2 \\ &= \sum_{e_k \in L_\lambda} \frac{d}{dx_\lambda} \left(\frac{b_{\lambda k} x_\lambda + \xi_k}{c_k} \right)^2 \\ &= 0, \end{aligned} \tag{16}$$



Initialize
Calculate TEF based on current $y_\lambda(t)$
Write the value of TEF on TEFM
Send TEFM to $\mathbb{L}_\lambda{}^a$
Receive TEFM
Received TEFM form all of $\mathbb{L}_\lambda{}^a$?
No
Yes
Compare TEF $\Phi(L_\lambda)$ with TEFs of $\mathbb{L}_\lambda{}^a$
$\Phi(L_\lambda)$ is the largest TEF ?
No
Yes
TF $\zeta(L_\lambda) = 0$
TF $\zeta(L_\lambda) = 1$
Check TF $\zeta(L_\lambda)$
TF $\zeta(L_\lambda) = 1$?
No
Yes
Conduct DATFO
TF $\zeta(L_\lambda) = 0$
Notify $\mathbb{L}_\lambda{}^a$ of TF $\zeta(L_\lambda)$ =0
Receive TF
Received TFS form all of $\mathbb{L}_\lambda{}^a$?
No
Yes
Confirm every TF of $\mathbb{L}_\lambda{}^a$ is 0

Fig. 4. Autonomous distributed control in tie-set (ADCT).

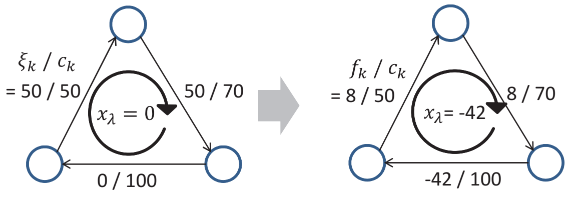


Fig. 5. Example for DATFO.

i.e.,

$$x_\lambda = -\frac{\sum_{e_k \in L_\lambda} b_{\lambda k}\xi_k / c_k^2}{\sum_{e_k \in L_\lambda} 1/c_k^2}, \tag{17}$$

since ${b_{\lambda k}}^2 = 1$.

In STEP 2, each edge flow f_k on $e_k \in L_\lambda$ is updated to $b_{\lambda k}x_\lambda + \xi_k$ where edge flows on L_λ are optimized to minimize the TFO function $\phi(x_\lambda)$ of L_λ.

Algorithm 1 Decentralized Algorithm for Tie-set Flow Optimization (DATFO)

```
STEP 0:
  Initialize a current tie-set flow $x_\lambda$ of $L_\lambda$ as 0.
  Get current flows $\Xi^\lambda$ of $L_\lambda$ provided by MV $y_\lambda(t)$ of TA.
STEP 1:
  Calculate $x_\lambda$ according to (17).
STEP 2:
  for each $f_k$ on $e_k \in L_\lambda$ do
    Update an edge flow $f_k$ to $b_{\lambda k}x_\lambda + \xi_k$.
  end for
```

An example of DATFO is shown in Fig. 5. In Fig. 5, the value of the TFO function before conducting DATFO is $\phi(x_\lambda) = 1.510$, and the value after DATFO is $\phi(x_\lambda) = 0.215$.

V. Simulation and Experiments

To verify the proposed method and analyze its performance as well as to solve the NFO problem described in section III, a simulator has been implemented in Java. To consider a real network's size and structure, the topology of the simulation network is based on ARPANET network (see Fig.6 of [17]), which has a bi-connected graph $G = (V, E)$ with 20 nodes and 32 links. The number of tie-sets is 13 and the height of a tree is 5, where the tree is created by Dijkstra's algorithm giving random costs on links. We use Common Buffering Method and Polling Method in a node. Each link capacity c_k is randomly assigned as $50 \le c_k \le 100$ $(k = 1, 2, ..., m)$. Every ties-set conducts ADCT with the time interval of 1 ms, and MV $y_\lambda(t)$ is constantly sent to a leader node of each tie-set. Experiments are conducted 20 times for each simulation below and we calculated the average value to capture general behavior.

A. Optimized Flows in ARPANET Network

We assign initial flows on a loop-free path from the given source s to sink t where the number of links on the path is 10. The value of each initial flow is uniformly assigned as $|\xi_k| = 50$. Experimental data are taken when every tie-set satisfies $-1 \times 10^{-3} < \frac{d}{dx_\lambda}\phi(x_\lambda) < 1 \times 10^{-3}$.

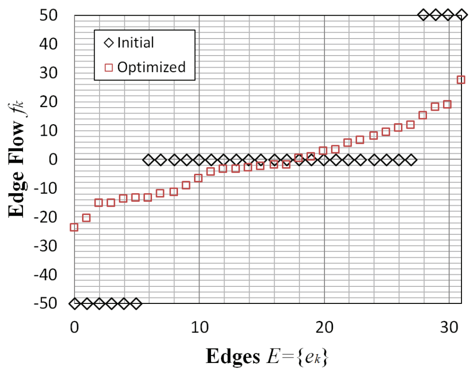


Fig. 6. Optimized edge flows in the ARPANET network with 20 nodes and 32 edges.

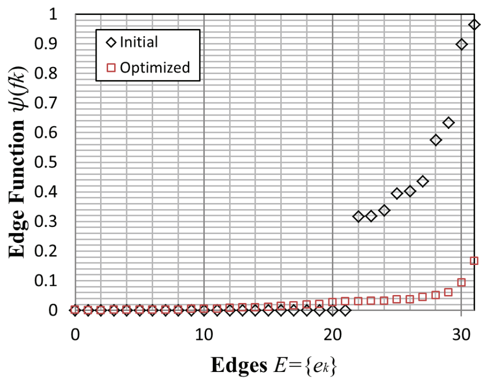


Fig. 7. Optimized values of edge functions in the ARPANET network with 20 nodes and 32 edges.

Fig. 6 and Fig. 7 respectively show the value of the edge flow f_k and the edge function $\psi(f_k)$ on each $e_k \in E$ before and after optimization in the ARPANET network. Those data are sorted in ascending order. As in the results shown in Fig. 6 and Fig. 7, all the edge flows are allocated in a balanced manner so that the value of the edge function of every link is minimized.

In this experiment, the network flow function Φ_N before and after optimization is $\Phi_N = 5.279$ and $\Phi_N = 0.774$, respectively. The time to complete the optimization is 45 (ms). The total number of computations for ADCT and DATFO is 421 and 72, respectively.

Aside from the ARPANET network, we also conducted experiments in randomly created 100-, 200-, 300-, and 400-node networks assuring their bi-connectivity. Those results are

TABLE II
Experiments in Networks with 100, 200, 300, 400 Nodes

$(\|V\|, \|E\|)$	$(100, 192)$	$(200, 392)$	$(300, 592)$	$(400, 792)$
Initial Φ_N	5.286	5.047	5.141	4.995
Optimal Φ_N	0.519	0.352	0.417	0.486
Time (ms)	225.50	226.75	599.00	1275.00

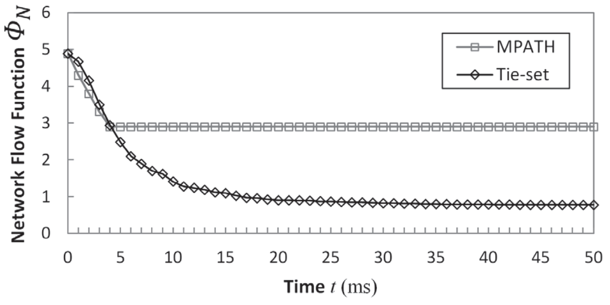


Fig. 8. Convergence of Network Flow Function Φ_N by Tie-set and MPATH based computations from time $t = 0$ to 50 (ms).

shown in TABLE II. From TABLE II, the optimized values of Φ_N in those networks show more than 90% reduction of initial Φ_N on average. This result indicates the radical improvement of throughput over an entire network. Although the time to complete optimization shows some quadratic increase, it is still feasible within a few seconds.

B. Comparison of Convergence Behavior with MPATH

In this experiment, we analyze the convergence behavior of the proposed optimization technique and compare it against Multiple Path computing method, which is often called MPATH, using the ARPANET network. MPATH provides multiple paths between each source-sink pair that need not necessarily have equal costs and that are loop-free at every instant (see [5]-[7] in detail). Simulation conditions are the same as those of the previous experiment.

As indicated in Fig. 8, MPATH converges faster than the proposed method as MPATH calculates different paths instantly and allocates flows on those paths. On the other hand, the proposed method based on tie-sets takes longer to converge on the optimal value than MPATH as it requires iterative computation within a fundamental system of tie-sets. The convergence line by Tie-set demonstrates subtle decrease throughout the optimization time span of Fig 8, whereas the network flow function Φ_N by MPATH does not change once it converges. The iterative computation based on tie-sets eventually realizes $\frac{d}{dx_\lambda}\phi(x_\lambda) \to 0$ so that the performance of the proposed method shows superiority over distributing flows by MPATH after $t \approx 5$ (ms). This result substantiates the optimization based on Tie-set performs better than MPATH once the network flows of an entire network are optimized.

VI. Conclusion and Future Work

In this paper, a network flow optimization (NFO) problem is first formulated with a tie-set flow vector space created on the basis of a fundamental system of tie-sets that represents a set of independent loops that underlie a mesh network. Then, we proposed a decentralized algorithm based on Tie-set to solve NFO problem where the computation to calculate optimal flows in a tie-set is iterated among the system of tie-sets. Simulation results in a 20-node network based on ARPANET and randomly created networks with 100 to 400 nodes demonstrate optimal flow allocations so that total throughput of an entire network has radically improved. The comparison experiment against the multiple path (MPATH) technique for flow distribution also shows the superiority of the proposed method with a modest delay in convergence.

The OpenFlow project [18] aims to include more flexible flow and path computation so that many optimization techniques can be implemented with open source environment. Therefore, we will integrate our decentralized algorithm into the OpenFlow Controller as a practical application of the proposed method.

Acknowledgment

The authors would like to thank researchers at Open Network Research Center (ONRC) at Stanford University for their invaluable comments and suggestions.

References

[1] S. Even, R. Tarjan, *Network flow and testing graph connectivity*, Princeton University Press, Princeton, NJ, USA 1975.
[2] T. Fujisawa, *Maximal flow in a lossy network*, Proc. of First Annual Allerton Conference on Circuit and System Theory, 1963, pp. 385 - 393.
[3] D. R. Ford, D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, USA 1962.
[4] M. Iri, I. Shirakawa, Y. Kajitani, S. Shinoda, etc, *Graph Theory with Exercises*, CORONA Pub: Japan, 1983.
[5] S. Vutukury, J.J. Garcia-Luna-Aceves, *MPATH: A Loop-free Multipath Routing Algorithm*, Elsevier Journal of Microprocessors and Microsystems, 2000, pp. 319–327.
[6] J. Chen, P. Drushel, D. Subramanian, *An efficient multi-path forwarding method*, Proc. of IEEE INFOCOM, 1998, Vol.3, pp. 1418-1425.
[7] J. Chen, P. Drushel, D. Subramanian, *A simple, practical, distributed multi-path routing algorithm*, Technical Report No. 98-320, Rice University, 1998.
[8] Xiao X, Hannan A, Bailey B, Ni LM, *Traffic engineering with MPLS in the Internet*, IEEE Networks 2000, 14(2):2833.
[9] Y. Lee, Y. Seok, Y. Choi, *A constrained multipath traffic engineering scheme for MPLS networks*, IEEE International Conference on Communications (ICC), 2002, pp. 2431-2436.
[10] S. Yongho, Y. Lee, Y. Choi, C. Kim, *Dynamic constrained multipath routing for MPLS networks*, Proc. of IEEE International Conference on Computer Communications and Networks, 2001, pp. 348-353.
[11] P. P. Herrmann, *On reducibility among combinatorial problems*, Report No TR-113, Project MAC, Massachusetts Institute of Technology, Cambridge, MA, 1973.
[12] N. Shinomiya, T. Koide, H. Watanabe, *A theory of tie-set graph and its application to information network management*, International Journal of Circuit Theory and Applications 2001; 29:367-379.
[13] T. Koide, T. Kubo, H. Watanabe, *A study on the tie-set graph theory and network flow optimization problems*, International Journal of Circuit Theory and Applications 2004, 32:447-470.
[14] J. Malinowski, *A new efficient algorithm for generating all minimal tie-sets connecting selected nodes in a mesh-structured network*, IEEE Transactions on reliability 2010; 59(1): 203 - 211.
[15] K. Nakayama, N. Shinomiya, H.Watanabe, *An autonomous distributed control method for link failure based on tie-set graph theory*, IEEE Transactions on Circuits and Systems-1, Regular Paper, Vol. 59, No. 11, 2012, pp. 2727 - 2737.
[16] K. Nakayama, K. Benson, V. Avagyan, M. Dillencourt, L. Bic, N. Venkatasubramanian, *Tie-set Based Fault Tolerance for Autonomous Recovery of Double-Link Failures* Proc. of IEEE Symposium on Computers and Communications (ISCC), July, 2013.
[17] M. Mdard, etc., *Generalized Loop-Back Recovery in Optical Mesh Networks*, IEEE/ACM Transactions on Networking, Vol. 10, No. 1, Feb. 2002, pp. 153 - 164.
[18] N. McKeown, etc., OpenFlow: Enabling Innovation in Campus Networks, ACM SIGCOMM Computer Communication Review, Vol. 38, Issue 2, pp.6974, April 2008.