論　文

An Optimal Routing Algorithm for Large Data Networks

Sung Woo Park*, Young Chon Kim** *Regular Members*

대규모 데이타 네트워크를 위한 최적 경로 설정 알고리즘

正會員 朴 聖 宇* 正會員 金 永 川**

ABSTRACT

For solving the optimal routing problem (ORP) in large data networks, an algorithm called the hierarchical aggregation/disaggregation and decomposition/composition gradient projection (HAD-GP) algorithm is proposed. As a preliminary work, we improve the performance of the original iterative aggregation/disaggregation GP (IAD-GP) algorithm introduced in [7]. The A/D concept used in the original IAD-GP algorithm and its modified version naturally fits the hierarchical structure of large data networks and we would expect speed-up in convergence. The proposed HAD-GP algorithm adds a D/C step into the modified IAD-GP algorithm. The HAD-GP algorithm also makes use of the hierarchical-structured topology of large data networks and achieves significant improvement in convergence speed, especially under a distributed environment. The speed-up effects are demonstrated by the numerical implementations comparing the HAD-GP algorithm with the (original and modified) IAD-GP and the ordinary GP (ORD-GP) algorithms.

要 約

대규모 데이타 네트워크에서 최적 경로 설정 문제를 해결하기 위해 HAD-GP (hierarchcal aggregation/disaggregation and decomposition/composition gradient projection) 알고리즘이 제안된다. 이를 위해 우선 [7]에서 제안된 IAD-GP (iterative aggregation/disaggregation GP) 알고리즘의 성능을 향상시킨다. 원래의 IAD-GP 알고리즘과 변형된 IAD-GP 알고리즘에서 사용된 A/D 개념은 본질적으로 대규모 데이타 네트워크의 계층적 구조에 적합하기 때문에 알고리즘의 수렴에 있어 속도 향상을 기대할 수 있다. 제안된 HAD-GP 알고리즘 역시 대규모 데이타 네트워크의 계층 구조화된 토폴로지를 이용하여, 특히 분산화된 환경하에서 수렴속도의 현저한 향상을 달성할 수 있다. 이 속도 향상 효과는 컴퓨터 모의 실험에 의해 HAD-GP 알고리즘을 IAD-GP와 일반적인 GP (ORD-GP) 알고리즘과 비교함으로써 보여진다.

* 한남대학교 정보통신공학과
Dept. of Information and Communication Eng. HanNam University
** 전북대학교 컴퓨터공학과
Dept. of Computer Eng. ChonBuk National University
論文番號 : 94-25

I. Introduction

Since optimization theory was introduced into the area of data networks, several efforts have been made to solve routing problems using those techniques [1]. The gradient projection (GP) algorithm [2] [3] is known to be one of the most popular and efficient optimization techniques applied to optimal routing problems (ORPs). Bertsekas *et al.* successfully implemented the GP algorithm for path-formulated optimal routing [4] and provided the validity and robustness of the GP algorithm in somewhat realistic environment [5] [6]. Tsai *et al.* also proposed an algorithm called the IAD-GP incorporating the existing GP algorithm and the *iterative aggregation/disaggregation* (A/D) method to solve the ORP for large data networks [7]. The main idea of the IAD-GP algorithm is to aggregate a physical network by a gateway-connected network whose disaggregated routing solution converges to the optimal one.

First of all, we propose a modified version of the IAD-GP algorithm which uses a slightly different A/D step from the original one. However, the performance of both the original and the modified IAD-GP algorithms is limited by the lack of detailed routing information on the gateway-connected network. Since the gateway-connected network adjusts flows between gateways/clusters, no detailed routing information for intra-cluster is used and this may cause the IAD-GP algorithm to slow down if the main bottle-neck in routing is caused by intra-cluster flows.

To overcome this problem due to unbalanced traffic flows, we propose another algorithm called the HAD-GP (hierarchical aggregation/disaggregation and decomposition/composition gradient projection) algorithm, which combines the modified IAD-GP algorithm with the *decomposition/composition* (D/C) method. In addition to running the modified IAD-GP routine, the HAD-GP algorithm runs a D/C step that decomposes a given network into several subnetworks, runs GP iterations on each subnetwork independently, and composes the routing solutions from those subnetworks. By allowing each subnetwork to run its individual GP iterations, the D/C step adjusts flows inside each subnetwork for which the A/D step cannot handle. This concept of *divide and conquer* fits the hierarchical structure of network topology and is well suited for distributed computation. Thus we expect a proper combination of the A/D and D/C steps with GP iterations to provide significant improvement in the speed of convergence.

This paper is organized as follows. In Section 2, we introduce two network models for the A/D and D/C steps and formulate the ORP with the GP algorithm as a solution technique. Section 2 discusses the speed-up effects achieved by using those network models. In Section 3, the details of the original IAD-GP algorithm and its modified version are presented. Section 4 proposes the HAD-GP algorithm. Performance comparison of the proposed algorithms (modified IAD-GP and HAD-GP) with the exisiting ones (ORD-GP and original IAD-GP) is provided through implementation results in Section 5. Finally, Section 6 summarizes this paper.

II. Problem Formulation

2.1 Network Models

Suppose that we are given a network with a large number of nodes. The network can be naturally transformed into a hierarchical network by clusterization. Each cluster at the k-th level is called the *level-k cluster*. A network consisting of these level-k clusters is also called the *level-k network*. Then what would the level-k network look like when viewed from higher $(k+1)$ level or lower $(k-1)$ level? Those views would provide the transformed networks which are different from the original level-k network. A higher level-$(k+1)$ cluster would not see the inside details of the level k clusters but consider them as ordinary nodes. On the other hand, a lower level-$(k-1)$ cluster recognizes only their peer level-$(k-1)$ clusters inside the level-k cluster they belong to.

These transformed networks generated from the view-point of higher level and lower level are called the level-k *aggregated* network and the level-k *decomposed* network, respectively. The level-k network itself is also called the level-k *detailed* network. Since the above argument can be generalized to any other levels, we drop out the term *level-k* and restrict our concerns to a 2-level hierarchical network hereafter.

Suppose that a given detailed network can be naturally partitioned into clusters. An aggregated network is created from the detailed network through an A/D step. The first step is to pick up the gateways of the detailed network with direct links between gateways unchanged. If there exists no direct link between gateways, a virtual link is provided. From the gateway-connected network abtained, we complete to construct the aggregated network by adding one virtual node per each cluster. The virtual node represents all the origins and the destinations inside the cluster and is fully connected to and from the gateways within the cluster.

A decomposed network is generated from a detailed network via a D/C step. The decomposed network corresponds to a cluster of the detailed network. Each cluster is directly transformed into a decomposed network by disconnecting all the inter-cluster trunks, so that several decomposed networks are usually created. While an A/D step transforms routing information to and fro between the detailed network and the aggregated network, a D/C step transforms routing information to and fro between the detailed network and the decomposed networks.

The above process of network transformation through the A/D and D/C steps are described in Figure 1. The detailed network contains eight nodes and can be divided into two clusters. Each cluster includes four nodes, two of which are used as gateways. The aggregated network consists of four gateways and two virtual nodes representing each cluster. For each virtual node, two virtual links connected to each gateway are also created. On the other hand, each cluster can be directly transformed into the decomposed network by disconnecting inter-cluster trunks.

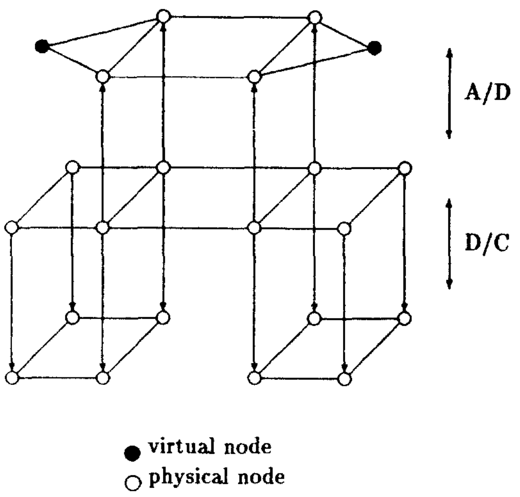

Figure 1. A/D and D/C steps.

2.2 GP method for ORP

Suppose that we are given a directed graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes, and $\mathcal{L}$ is the set of directed links. Let W be a set of origin-destination (OD) pairs, and for each OD pair $w \in W$, let the traffic demand be r_w. The main variable is the set of path flows $x = \{x_p\}$ which satisfies the following constraints :

$$\sum_p x_p = r_w, \quad x_p \geq 0, \qquad p \in P_w, \ \forall w \in W \qquad (1)$$

where P_w is a given set of paths for an OD pair w. Consider a link cost function :

$$D_{ij}(F_{ij}) = \frac{F_{ij}}{C_{ij} - F_{ij}}$$

where C_{ij} is the capacity of link $(i, j) \in \mathcal{L}$ and F_{ij} is the flow on link $(i, j) \in \mathcal{L}$. $D_{ij}(F_{ij})$ indicates the average number of data packets waiting or being transmitted on link $(i, j) \in \mathcal{L}$ assuming the behavior of an M/M/1 queue in equilibrium. The overall cost function is defined by the sum of each link cost function divided by the total traffic demands over the network :

$$D(x)=\frac{1}{\gamma}\sum_{(i,j)\in \mathcal{L}} D_{ij}(F_{ij})$$
$$=\frac{1}{\gamma}\sum_{(i,j)\in \mathcal{L}} D_{ij}\left[\sum_{p\in P_{ij}} x_p\right]$$

where P_{ij} is the set of all paths traversing link $(i, j)\in \mathcal{L}$ and $\gamma=\sum_{w\in W} r_w$. The path-formulated ORP is that of minimizing $D(x)$ subject to the constraints (1).

Define a variable s_w to be the path having the minimum first derivative length (MFDL) and x_{s_w} to be its flow for w. The FDL and the MFDL on path p are given by

$$d_p=\sum_{(i,j)\in l_p} D'_{ij}(F_{ij}),\quad d_{s_w}=\min_p d_p,\quad p\in P_w,\ w\in W$$

where l_p is the set of links belonging to a path p. By substituting $x_{s_w}=r_w-\sum_{p\in P_w,\ p\neq s_w} x_p,\ \forall w\in W$ into $D(x)$, we obtain a transformed form of ORP involving only positivity constraints :

$$\text{minimize}\qquad \tilde{D}(\tilde{x})$$
$$\text{subject to } x_p\geq 0,\ \ p\in P_w,\ p\neq s_w,\ \forall w\in W.$$

$\tilde{D}(\tilde{x})$ now becomes a function of $\tilde{x}=\{x_p\}$, $p\in P_w$, $p\neq s_w$, $\forall w\in W$. The GP iterations for the ORP take the following form :

$$x_p^{k+1}=\max\{0,\ x_p^k-\alpha^k(H_p^k)^{-1}(d_p^k-d_{s_w}^k)\},$$
$$p\in P_w,\ p\neq s_w,\ \forall w\in W \qquad (2)$$

where k indicates the current iteration step and α is a step size. The second derivative length (SDL) is given by

$$H_p=\sum_{(i,j)\in L_p} D''_{ij}(F_{ij})$$

where L_p is the set of links belonging to either path p or s_w, but not both. The GP iterations (2) is called the detailed GP when applied on the detailed network. For the aggregated and the decomposed networks, they are also called the aggregated GP and the decomposed GP, respectively.

The convergence proof of the detailed GP algorithm is well described in [8]. Since each of the aggregated and the decomposed networks can be considered as another non-hierarchical detailed network, it is easy to see that the convergence results derived in [8] can be extended for solving the ORP in those networks.

2.3 Speed-Up Effects

Suppose that the aggregated and the decomposed GP iterations are performed in a distributed way. The computation time of a GP iteration is dominated by the shortest path routine involved whose time complexity depends on the number of nodes in the network. Noting that the transformed networks consist of relatively smaller number of nodes than the detailed network, it is expected that the convergence of the aggregated and decomposed GP iterations can be accelerated.

For an example of the speed-up effects, consider an n-node detailed network consisting of c-clusters. Assume also that each cluster has a finite number of gateways bounded by $O(c)$. Dijkstra algorithm is chosen for the shortest path computation of the GP iteration. Then each node is responsible for finding the shortest paths for all OD-pairs with itself as the origin. The time complexity of Dijkstra algorithm is $O(n^2)$ for an n-node network. Since the detailed network consists of n nodes, it takes $O(n^2)$ times for every node to find the shortest paths for all other destinations. On the other hand, Dijkstra algorithm running on the aggregated network solves the shortest path problem within $O(c^2)$ times since the aggregated network has only $O(c)$ number of nodes. Computation time per GP iteration now can be saved by the difference of $O(n^2-c^2)$.

With the decomposed networks, similar type of time-savings for routing computation can be expected. Consider again an n-node detailed network with the number of clusters bounded by $O(c)$. This time the detailed network is transformed downward into several c-decomposed networks of size $O(\frac{n}{c})$. Dijkstra algorithm is also applied to find the shortest paths between all OD pairs in a distributed way. Then it takes $O(\frac{n^2}{c^2})$ times for each of the decomposed network to find the shortest paths of all OD pairs. Comparing with $O(n^2)$ times of the detailed network, the computation time per GP iteration can be again saved by the order of c^2 folds.

From another point of view, the GP algorithm converges in $O(h|W|P_{max})$ number of iterations [9], where h is the diameter of the network, $|W|$ is the total number of OD pairs, and P_{max} is the maximum number of active paths per OD pair for all iterations and all OD pairs. For the aggregated network, h usually redeces by the factor of $O(c)$ and $|W|$ by the factor of $O(c^2)$. For the decomposed networks, h reduces by $O(\frac{n}{c})$ and $|W|$ by $O(\frac{n^2}{c^2})$. Accordingly, when applied for the aggregated and the decomposed networks, the overall time complexity of the GP algorithm reduces by the factor of $O(c^3)$ and $O(\frac{n^3}{c^3})$, respectively.

Ⅲ. IAD-GP Algorithm

The A/D step used in the IAD-GP algorithm consists of three substeps : aggregation, GP iterations and disaggregation. Once an aggregated network is generated from a detailed network, the aggregated GP iterations are successively performed until the convergence is reached. After then,. the aggregated network is disaggregated into the detailed network and the control of algorithm returns back to the main routine where the detailed GP iterations are applied.

3.1 A/D step [7]

The A/D step starts by constructing an aggregated network from a given detailed network. Then the detailed path flows sharing the common gateways are aggregated :

$$x_{\hat{p}} = \sum_{p \in \hat{p}} x_p, \quad p \in P_w, \forall w \in W, \hat{p} \in \hat{P}_{\hat{w}}, \forall \hat{w} \in \hat{W}$$

where $\hat{P}_{\hat{w}}$ represents the set of aggregated path for each aggregated OD pair $\hat{w} \in \hat{W}$. The FDL and the SDL, which are essential to the aggregated GP iterations, are given by

$$d_{\hat{p}} = \frac{\sum_{p \in \hat{p}} x_p d_p}{x_{\hat{p}}}, \quad d_{\hat{p}\hat{q}} = \frac{\sum_{p \in \hat{p}} \sum_{q \in \hat{q}} x_p x_q d_p}{x_{\hat{p}} x_{\hat{q}}},$$

$$p \in P_w, q \in Q_w, \forall w \in W, \hat{p} \in \hat{P}_{\hat{w}}, \hat{q} \in \hat{Q}_{\hat{w}}, \forall \hat{w} \in \hat{W}$$

where $Q_w(Q_{\hat{w}})$ is another set of active paths belonging to $w(\hat{w})$ and may or may not correspond to $P_w(P_{\hat{w}})$. Aggregated paths $\hat{p}$ and $\hat{q}$ must be mutually disjoint. For successive running of the aggregated GP iterations, aggregated virtual link capacity (AVLC) on link (i, j), denoted by $\hat{C}_{ij}$, is estimated using the FDL of link delay with respect to the flow :

$$d_{\hat{p}} = \frac{\hat{C}_{ij}}{(\hat{C}_{ij} - \hat{F}_{ij})^2}, \quad (i, j) \in \hat{\mathfrak{L}}, \hat{p} \in \hat{P}_{\hat{w}}, \forall \hat{w} \in \hat{W} \quad (3)$$

where $\hat{\mathfrak{L}}$ is the set of links on the aggregated network and $\hat{F}_{ij} = x_{\hat{p}}$.

Now the aggregated GP iterations are obtained from (2) by replacing the detailed path flow x_p with the aggregated path flow $x_{\hat{p}}$:

$$x_{\hat{p}}^{k+1} = \max\{0, x_{\hat{p}}^k - \alpha^k (d_{\hat{p}\hat{q}_{\hat{w}}}^k)^{-1} (d_{\hat{p}}^k - d_{\hat{p}_{\hat{w}}}^k)\},$$

$$\hat{p} \in \hat{P}_{\hat{w}}, \hat{p} \neq \hat{p}_{\hat{w}}, \forall \hat{w} \in \hat{W}.$$

The disaggregation procedure properly distributes the aggregated path flows back into the detailed path flows. It disaggregates each aggregated path flow according to a weighted average of the corresponding detailed path flows. The weight is determined by the amount of the detailed path flow assigned at the previous detailed GP iteration (before aggregation) :

$$x_p = \frac{x_p}{\sum_{p \in \hat{p}} x_p} \cdot x_{\hat{p}},$$

$$p \in P_w, \ \forall w \in W, \ \hat{p} \in \hat{P}_{\hat{w}}, \ \forall \hat{w} \in \hat{W}.$$

3.2 Modified Version

Recalling the eq. (3), in the original A/D step, the AVLC is estimated using the initial aggregated path flow and remains fixed while running the aggregated GP iterations. The fixed AVLC implies that the dynamic situations inside the cluster cannot be taken into account in performing the aggregated GP iterations. Thus the performance of the aggregated GP iterations is restricted as to distributing the aggregated path flows optimally. As a result, more detailed GP iterations may be required to obtain the global optimal routing solution after disaggregation. It is possible to perform the A/D step for every aggregated GP iteration. In this case, however, processing time to switch the detailed network to and from the aggregated network may not be ignored.

The original disaggegation procedure also has a drawback. Flows on the detailed paths can be reassigned only when they had existed before the A/D step was applied. Suppose that an aggregated path is created after the A/D step and is to be disaggregated into the detailed paths. It turns out that the original disaggregation scheme fails unless there exists any detailed path having nonzero flow before aggregation ; the completeness condition is not satisfied.

The above arguments state the A/D step must be carefully inserted between any two successive detailed GP iterations. For the original A/D step to work well, the detailed network should have been to some extent stabilized before aggregation. However, this contradicts the purpose of using the A/D step to reduce the convergence time in large networks. The A/D step becomes more effective as early as they are inserted since the aggregated GP iterations are allowed to shift larger amount of path flow in the direction of the optimality than the detailed GP iterations. To overcome these problems, we devise more efficient schemes for the AVLC and the disaggregation. With these schemes, the A/D step can be inserted much earlier than the original A/D step.

Although the virtual links between gateways inside each cluster provide transit paths for inter-cluster path flows, this effect can be ignored provided that traffic demands are easily distributed over the clusters. That is, the AVLCs in each cluster are assumed to be sufficient to accommodate traffic demands such that the main part of cost function arises from intra-cluster traffic flows. For this purpose, the AVLC denoted by $\hat{C}_{ij}^{max}$ takes the maximum amount of flow that the attached gateways can handle on its outgoing/incoming links :

$$\hat{C}_{ij}^{max} = \sum_{(i,k)\in \mathcal{L}_i} C_{ik}, \quad (i,j) \in \hat{\mathcal{L}}, \ (i,k) \in \mathcal{L}_i$$

where $\mathcal{L}_i \subset \mathcal{L}$ is the set of outgoing links attached to gateway i. The AVLC only needs be large enough not to cause any overflow on the virtual link. Once link overflow happens, it takes many of the aggregated GP iterations to recover from it – much more than in normal situation.

The disaggregation scheme is now based on a weighted average of the current aggregated path flows. We first compute the ratio of each aggregated path flow to the traffic demand of an aggregated OD pair and use this ratio for distributing the aggregated path flows into the detailed path flows. That is, each detailed path is reassigned the path flow depending on the flow ratio determined by the corresponding aggregated path flow, not by the previous detailed path flow. If no pre-existing detailed path is found, we find the subpaths with MFDL from each cluster and combined them to form a new detailed path. The MFDL subpaths are generated by running a shortest path routine on each cluster. This disaggregation scheme produces at least one detailed path for every aggregated path :

$$x_p = \frac{x_{\hat{p}}}{\sum_{\hat{p} \ni p} x_{\hat{p}}} \cdot r_{\hat{w}},$$

$$p \in P_w, \ \forall w \in W, \ \hat{p} \in \hat{P}_{\hat{w}}, \ \forall \hat{w} \in \hat{W}.$$

Ⅳ. HAD-GP Algorithm

By transforming a detailed network into an aggregated network, the IAD-GP algorithm reduces the problem size (number of nodes) to speed-up the convergence rate. For the same purpose, the detailed network can be divided into a set of decomposed networks (clusters). The decomposed GP iterations are applied on each cluster in a separate and distributed way. After then, the control of the algorithm returns to the detailed network with the decomposed routing solutions combined from each cluster. This procedure is called the D/C step.

The decomposed networks have intra-cluster information only, where as the aggregated network has inter-cluster information. By alternately running the A/D and D/C steps, the HAD-GP algorithm has the freedom to adjust path flows in two distinct (inter-cluster and intra-cluste) components of routing. That is, the HAD-GP algorithm seeks the approximate optimal routing solution through the proper combinations of the A/D and D/C steps while saving computation time. Finding the exact optimal routing solution will be left to the detailed GP iterations.

4.1 Virtual Capacity

Since the decomposed networks are available, the A/D step of the HAD-GP algorithm is capable of estimating the congestion level of each decomposed network. The congestion level is determined by the link utilization averaged over the decomposed network. Now the average link utilization of each decomposed network is used for deciding the AVLC. The average link utilization is initially computed by running the detailed GP iterations only with intra-cluster traffic demands. Then a new AVLC is given by

$$\hat{C}_{ij} = \hat{C}_{ij}^{max}(1 - \check{\rho}), \quad (i, j) \in \hat{\mathcal{L}}$$

where $\check{\rho}$ is the average link utilization computed from intra-cluster path flows. That is, the AVLC in the HAD-GP algorithm is the maximum available capacity of the gateways from which a portion of optimal intra-cluster path flows are subtracted.

4.2 D/C step

For decomposition, a detailed network is simply partitioned into independent clusters. Each detailed path is also partitioned into subpaths according to the decomposed networks involved. For an intra-cluster path, a detailed path is exactly equivalent to the corresponding decomposed path. For an inter-cluster path, however, a detailed path is broken into several decomposed paths cluster by cluster. The decomposed path flow $\check{p}_i$ in a decomposed network i is expressed by

$$x_{\check{p}_i} = \sum_{p \supset \check{p}_i} x_p, \quad p \in P_w, \ \forall w \in W, \ \check{p}_i \in \check{P}_{\check{w}_i} \ \forall \check{w}_i \in \check{W}_i$$

where $\check{P}_{\check{w}_i}$ represents the set of decomposed paths for a decomposed OD pair $\check{w}_i \in \check{W}_i$. That is, for all the detailed path flows sharing the common part in each cluster, a decomposed path is created and the sum of the detailed path flows is assigned as the decomposed path flow.

The decomposed GP iterations for the decomposed network i are obtained again from (2) by replacing the detailed path flow x_p with the decomposed path flow $x_{\check{p}}$:

$$x_{\check{p}_i}^{k+1} = \max\{0, x_{\check{p}_i}^{k} - \alpha^k (\check{d}_{\check{p}_i \check{p}_{\check{w}_i}}^{k})^{-1} (d_{\check{p}_i} - d_{\check{p}_{\check{w}_i}})\},$$

$$\check{p}_i \in \check{P}_{\check{w}_i}, \ \check{p}_i \neq \check{p}_{\check{w}_i}, \ \forall \check{w}_i \in \check{W}_i.$$

For composition, several decomposed paths in each cluster are combined to form a new detailed path. The crust of the composition scheme is to piece independently developed decomposed paths to generate new detailed paths. Let us denote a new detailed path after a D/C step by p' and its path flow by $x_{p'}$. A composition scheme to generate new detailed paths from independently developed decomposed paths is as follows :

```
Repeat (a)-(c) until $x_p = 0,\ p \in P_w,\ \forall w \in W$
    a) $p' = \{ \cup \check{p}_i : x_{p'} = \min[x_p, \min_i(\max_{\check{p}_i} x_{\check{p}_i})]\},$
       $\check{p}_i \in \check{P}_{\check{w}_i},\ \forall \check{w}_i \in \check{W}_i$
    b) $x_p = x_p - x_{p'}$
    c) $x_{\check{p}_i} = x_{\check{p}_i} - x_{p'},\quad \check{p}_i \in \check{p}'.$
```

Index i denotes the identity of the decomposed networks that the detailed path traverses before the D/C step is applied.

Each iteration of this composition procedure produces a detailed path. A careful examination of the proposed composition scheme indicates that the computational overhead is in the order of $|W|P_{max}$ where $|W|$ is the total number of OD pairs and P_{max} is the maximum number of active (detailed) paths per OD pair for all iterations and all OD pairs. Recall that the time complexity of the detailed GP algorithms is $O(h|W|P_{max})$ where h is the diameter of the network. Then the time complexity $O(|W|P_{max})$ of the proposed composition scheme is comparable to that of one detailed GP iteration.

Figure 2 describes an example of this composition step. Consider a detailed path traversing three clusters (Ⅰ, Ⅱ and Ⅲ)(Figure 2(a)). After partitioning the detailed path into three decomposed paths and applying the decomposed GP iterations independently, several new decomposed paths have been developed in each of the composed network: two paths in cluster Ⅰ, three paths in cluster Ⅱ and two paths in cluster Ⅲ (Figure 2(b)). Initial path flows of 7 are preserved in each decomposed network. For composition, the first maximum path flow common to all decomposed paths in each cluster is found to be 4 and is assigned to the detailed path composed of the corresponding decomposed paths. The second maximum path flow available is 2 and is assigned to another newly-created detailed path. The remaining decomposed paths naturally constitutes the third detailed path with the path flow of 1. As a result, three different new detailed paths are obtained from the decomposed paths (Figure 2(c)).

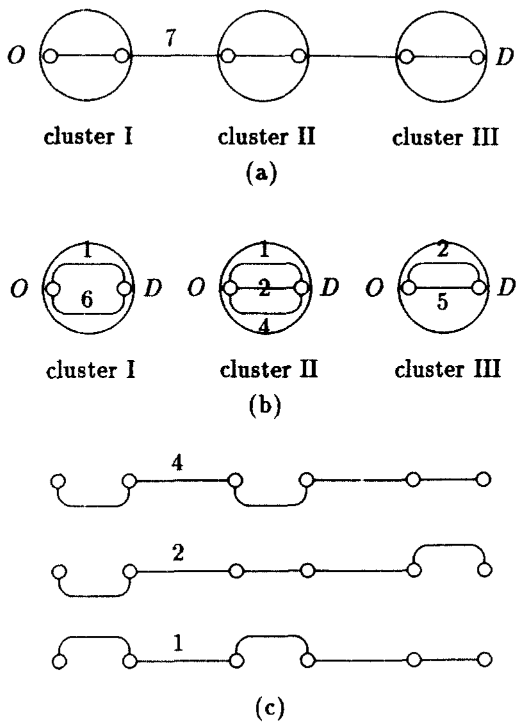

Figure 2. Composition procedure (a) A detailed path before decomposition. (b) Decomposed paths developed by the decomposed GP iterations. (c) Detailed paths after composition

Ⅴ. Implementation Results

In this section, the improvement of convergence speed achieved by the (original and modified) IAD-GP and the HAD-GP algorithms is demonstrated through implementation results. The source program is written in C and run on a SUN 4/370. Figure 3 shows a 52-node network partitioned into four decomposed networks. Every link has a capacity of 50 except that inter-cluster trunks have 80. Two sets of OD pairs are generated; one has 100 OD pairs and the other has 60 OD pairs. Each set randomly generates traffic demands ranging from 1 to 10 and 1 to 15 with the average of 4.92 and 7.62, respectively. The ratio of inter-cluster to intra-cluster OD pairs is 40/60 for the set of

100 OD pairs and is 40/20 for the set of 60 OD pairs.

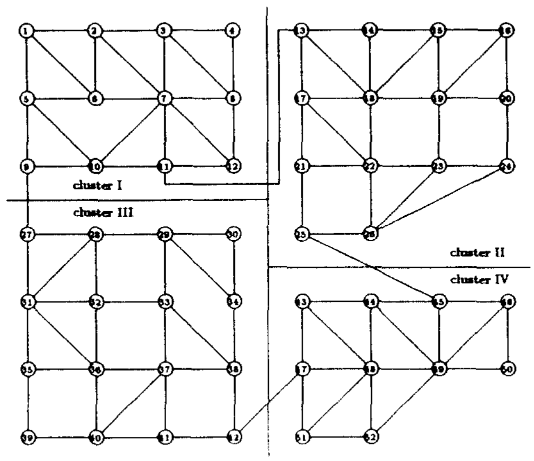

Figure 3. 52-node network.

Four algorithms (ORD-GP, original and modified IAD-GP, HAD-GP) are applied on the same network. For the HAD-GP algorithm, AVLCs for the A/D step is first estimated. Both the (original and modified) IAD-GP and the HAD-GP algorithms start directly with the A/D step without running the detailed GP iterations. The A/D step in the HAD-GP algorithm is followed by the D/C step. Then the control of both algorithms returns to the detailed GP iterations if necessary. A constant step size is used for all of four algorithms. We choose 0.1 for the detailed GP iterations, 1.0 for the (original and modified) aggregated GP iterations and 0.3 for the decomposed GP iterations. These values are the maximum allowable step size for each algorithm to avoid any undesirable behavior (oscillation or divergence etc.) The aggregated and decomposed paths are allowed to shift larger amount of flow at one time than the detailed paths. The larger step sizes used for the A/D and D/C steps are one of the main advantages by which the convergence of the aggregated and decomposed GP iterations can be accelerated.

Table 1 through Table 6 show the implementation results for each set of OD pairs generated (the ratio of inter-cluster to intra-cluster OD pairs are indicated inside the parenthesis). For performance comparison, we measure an average CPU time spent to run each algorithm, the number of iterations required for convergence and the final value of objective function (average network delay). For each measurement, we run each

Table 1. CPU time (40/60)

ORD-GP	original IAD-GP	modified IAD-GP	HAD-GP
14.0	11.2	9.8	3.6

Table 2. Number of iterations (40/60)

	ORD-GP	original IAD-GP	modified IAD-GP	HAD-GP
detailed GP	8	6	5	1
aggregated GP		2	3	3

disaggregated GP				7(cluster Ⅰ)
				6(cluster Ⅱ)
				8(cluster Ⅲ)
				7(cluster Ⅳ)

Table 3. Average network delay (40/60)

ORD-GP	original IAD-GP	modified IAD-GP	HAD-GP
54.91	54.95	54.94	55.55

Table 4. CPU time (40/60)

ORD-GP	original IAD-GP	modified IAD-GP	HAD-GP
84.2	63.3	55.5	4.3

Table 5. Number of iterations (40/60)

	ORD-GP	original IAD-GP	modified IAD-GP	HAD-GP
detailed GP	53	39	34	1
aggregated GP		7	9	9

disaggregated GP				7(cluster Ⅰ)
				7(cluster Ⅱ)
				11(cluster Ⅲ)
				27(cluster Ⅳ)

Table 6. Average network delay (40/60)

ORD-GP	original IAD-GP	modified IAD-GP	HAD-GP
72.30	72.43	71.54	76.03

of four algorithms 10 times and take the average on the results.

In terms of CPU time, the improvement of convergence speed by the HAD-GP algorithm ranges approximately from 4 to 20 folds over the ORD-GP algorithm while the (orginal and modified) IAD-GP algorithm improves only by 1.3 to 1.5 folds. (Table 1 and Table 4). From the above results, the HAD-GP algorithm is much more useful when the network is initially overflowed by the shortest path routine (Table 4 through Table 6). We also observe that the modified IAD-GP algorithm performs better than the original IAD-GP algorithm, especially when the network is initially overloaded. In this case, the original aggregated GP iterations converge faster than the modified aggregated GP iterations. Instead, the disaggregated routing solution from the original A/D step is farther away than that of the modified A/D step so that more detailed GP iterations are needed to reach the global optimality.

Let us consider savings of computation time in terms of the number of iterations. The main burden in a GP iteration is to find the shortest paths, taking $O(n^2)$ times. In Figure 3, the number of nodes in an aggregated network or a decomposed network is about one-forth the detailed network. Thus 16 aggregated and decomposed GP iterations correspond to one detailed GP iteration. Compared with the ORD-GP algorithm, the HAD-GP algorithm improves the convergence rate by up to 10 folds while the modified IAD-GP algorithm improves by 1.5 folds even considering the overhed due to the A/D and D/C steps (Table 5). The modified IAD-GP algorithm also outperforms the original one by approximately 4 detailed GP iterations.

The average network packet delays evaluated at the convergence are also shown in Table 3 and Table 6. The difference between any pair of the algorithms falls within 5% of each other. Thus, it can be said that the HAD-GP algorithm improves the convergence speed significantly without sacrificing the optimality provided that the parameters are properly chosen. Furthermore, if the HAD-GP algorithm is implemented in a distributed way, we can expect more drastic savings of computation time in terms of CPU time and the number of iterations. For example, the overall number of decomposed GP iterations in the HAD-GP algorithm reduces to 27 in Table 5 because we only consider the maximum number of iterations among the decomposed networks. The distributed HAD-GP algorithm would reduce again the number of iterations as well as the CPU time required for convergence.

VI. Summary

We have proposed the modified IAD-GP routing algorithm combining the GP algorithm with the A/D step. However, the (original and modified) IAD-GP algorithm turns out to be inefficient under certain situations ; for example, if a cluster is heavily congested and initially introduces some unbalanced traffics to the network, the A/D step does not provide the significant improvement of convergence speed. In addition, the disaggragated routing solution of the IAD-GP algorithm may be too far away from the global optimal routing solution. The main reason is that the A/D step only tries to solve the ORP from the view-point of inter-cluster path flows.

To overcome this problem, we proposed the HAD-GP algorithm combining the modified IAD-GP algorithm and the D/C step. The D/C step divides a large problem into a number of smaller-size subproblems and solves them separately. Solving those individual subproblems simultaneously can provide tremendous savings of computation time provided that the subproblems are loosely coupled. The HAD-GP algorithm views the routing problem from the higher level (A/D step) as well as the lower level (D/C step) and converges to the optimal routing solution faster than either the ORD-GP algorithm or the (original and modified) IAD-GP algorithm.

References

1. A. Ephremides and S. Verdú, "Control and optimization methods in communication network problems," *IEEE Trans. Automat. Contr.*, vol. AC 34, pp. 930-942, 1989.
2. D. P. Bertsekas, "On the Goldstein-Levin Polyak gradient projection method," *IEEE Trans. Automat. Contr.*, vol. AC-21, pp. 174-184, Apr. 1976.
3. D. P. Bertsekas and R. G. Gallager, *Data Networks*. Englewood Cliffs, NJ : Prentice Hall, 1987.
4. D. P. Bertsekas, B. Gendron, and W. K. Tsai, "Implementation of an optimal multicommodity network flow algorithm based on gradient projection and a path flow formulation," Mass. Inst. Technol., Combridge, MA, LIDS Rep., p-1364, 1984.
5. J. N. Tsitsiklis and D. P. Bertsekas, "Distributed asynchronous optimal routing in data networks," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 325-332, Apr. 1976.
6. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Englewood Cliffs, NI : Prentice-Hall, 1989.
7. W. K. Tsai, G. Huang, J. K. Antonio and W. T. Tsai, "Distributed iterative aggregation algorithms for box-constrained minimization problems and optimal routing in data networks," *IEEE Trans. Automat. Contr.*, vol. AC-34, pp. 34-46, 1989.
8. W. K. Tsai, "Convergence of gradient projection routing methods in an asynchronous stochastic quasi-static virtual circuit network," *IEEE Trans. Automat. Contr.*, vol. AC-34, pp. 20-33, Jan. 1989.
9. W. K. Tsai, "Complexity of gradient projection method for optimal routing in data networks," to appear in *IEEE Trans. Automat. Contr.*, 1991.

朴 聖 宇(Sung Woo Park) 정회원

1962년 9월 13일생

1981년 3월 ~ 1985년 2월 : 연세대학교 전자공학과 졸업(공학사)

1985년 1월 ~ 1986년 3월 : 한국 데이타 통신(주) 연구원

1986년 6월 ~ 1989년 8월 : Texas A&M Univ. Dept. of Electrical Engineering(MS)

1989년 9월 ~ 1991년 12월 : Univ. of Califonia, Irvine Dept. of Electrical and Computer Engineering(Ph.D)

1992년 3월 ~ 현재 : 한남대학교 정보통신공학과 조교수

※주관심분야 : Computer Networks

金 永 川(Young Chon Kim) 정회원

1956년 12월 10일생

현재 : 전북대학교 컴퓨터공학과 부교수

제18권 제1호 참조