Reverse Shortest Path for Dataflow Optimization of Hierarchically Structure Data Networks

PRAWAT CHAIPRAPA
The University of The Thai Chamber of Commerce
Department of Computer and Multimedia Engineering
126/1 Vibhavadee-Rangsit Road, Dindaeng,
Bangkok Thailand
prawat_cha@utcc.ac.th

KAJORNSAK KANTAPANIT
North Chiang-Mai University
Department of Computer Engineering
169 Moo 3 Nong Kaew, Chiang Mai
Thailand
kajornsak@eng.cmu.ac.th

Abstract: In the routing process to select the data paths for Hierarchically Aggregation/Disaggregation and Composition/Decomposition,(HAD) networks, a fast algorithm for finding optimum paths for dataflow is needed. In this research we propose an algorithm called the Reverse Shortest Path algorithm to improve the speed in the calculating procedure for finding the shortest paths. This algorithm performs the reversed calculation in stead of the forward calculation used in conventional algorithms. The demand in each original destination pair (OD pair) has been distributed to the sub OD pairs in each relevant subnetwork $r_{(u,v)} = r_{(u,l)} = ... = r_{(l,k)} = r_{(k,v)}$ with l and k, the gateways and ancestors in the active path. For each different commodities, the parallel processing is carried out with the shared shortest path processing time of $O(\log(n))$ which less than $O(m\log(n))$ of HAD algorithm[1] where, n is the number of nodes in the networks, M is the number of commodities in each cluster and m is a positive integer which is less than M. The proposed algorithms have been developed and tested on a simulated network of 200 nodes clustered into 20 groups. Each group uses a personal computer as the processor for the group. Ten data *Monte Carlo* simulation patterns were generated for the test. The first five patterns represent typical normal dataflows which largely consist of short distance communications. The other five patterns represent the worst case data communication scenario. Test results on the proposed Reverse Shortest Path algorithm show that, for the tested network, the algorithm has improves the speed in finding the shortest paths by 20% as compared to the conventional shortest path algorithm.

Key–Words: Optimal routing,distributed computation, gradient projection method, hierarchically structure network

1 Introduction

Various methods to solve the optimal routing problems including the methods using linear programming [20],[19],[21] which is suitable for the networks with small number of nodes, and the methods using non-linear programming for the network with larger number of nodes [10][11][4],[7],[8] and [9]. One of them is called the Projected Newton Method (PNM), it has been applied and investigated in [10] and [11]. Later on the Gradient Projection Method (GPM) has been proved to be much more efficient than the PNM method because in the the GPM method, the $H(k)$ which is the Hessian matrix of $X(k)$ has been taken into consideration giving a more converged solution than the PNM method. The main reason is that the (ORP) is the non-linear flow space problem in which the time spent to get the solution depends on three different factors, they are, the number of commodities, the number of shared links and the number of nodes in the network. Many researches have been carried on to speed up the processing time in dealing with these factors. In [4],[7],[8] and [9], the GPM with parallel processing according to the commodities giving the processing time of $O(M\Phi^2)$ where in [12], it has been analysed that M is the number of commodities or the number of the original nodes with the same OD pair and Φ is the diameter of the network. But, anyhow, for a large network with larger number of M, this method takes longer processing time. In [2],[13] and [14], the HAD network structure has been proposed to give the processing time of $O(M\log^2(n))$, but this method is suitable only for the “hierarchically clustered network topology” which does not conform to the general practical networks. In [15], a fast shortest path algorithm for the hierarchically clustered networks has been investigated and gives a better speed to find the solution for the ORP, but this method works fine for only the hierarchically clustered networks.

Since the method with parallel processing in accordance to the commodities is still not suitable for larger networks, many researchers proposed varieties of ideas to reduce the shortest path calculation by the distributed and parallel processing called the parallel textured algorithm [5],[6] giving the ORP processing time of $O(((1+\eta)n/m)^2$ with m the number of subnetworks, n the number of nodes in the network and η the ratio of the number of nodes in the overlapped path to the number of the nodes in the network.

Later on, in [1], a technique to combine the decomposition and composition (D/C) for the hierarchically structured network which is current internet network has been proposed to speed up the processing time with the parallel textured algorithm along with the hypothesis to configure each path to the destination to pass through a gateway. This method gives the processing time of $O(m\log^2(n)$. But in the process of this method, during the adjust path flow and $H(k)$ approximation process, the calculation for the shared parts of the links has to be done separately for each commodity resulting in the delayed time in the communications between processors.

It can be seen that in [1] as stated earlier, the ORP is to be solved at an improved speed by using paralleling processing for the popular internet which is a large network of hierarchically structure. The algorithm used was the GPM with two steps of processing. The first step was to find the shortest path for each OD pair and the second step was to adjust the path flow for each OD pair. But, in the process of distribution of parallel shortest path calculations there were the overlapped parts between processors and the processors had to spend the extra time of $O(m\log(n))$ to update the overlapped parts. We, then, propose an algorithm to reduce the update overlapped parts calculations by distributing the demand of each OD pair to be spread along the subpaths between many original nodes to the gateway with the same destination node and we use the reversed shortest path to eliminate the overlapped parts updating. This helps speed up to find the solutions for the ORP problems.

The next section give the details of the ORP formulation, the definitions and assumptions of the network are explained in section 3, the optimal routing algorithm is discussed in section 4, the test results are presented next and the conclusion is given in the last section.

2 ORP formulation of hierarchically structure networks

2.1 ORP Formulation

In this section, we begin with the standard ORP formulation [7][8] and formulation [1] which are different from [7][8] by the consideration of the flow of all the commodities at one instant. We are given a network $G=(V,L)$,where V is set of nodes,$|V|=N$ and L is a set of links and W is set of ordered pairs w of distinct nodes referred to as the origin and the destination of w. For each w, $(|W|=M)$. We are given a scalar r_w referred to as the flow demand of w. The ORP is formulated with the objective of dividing each r_w among the paths from the origin to the destination in such a way that the resulting total link flow pattern minimizes a cost function of interest. Let P_w denote a given set of paths that have the same origin and destination as w. And Let $x_{w,p}$ denote the flow of path p. The constraints follow:

$$x_{x,p}\geq 0, \forall p\in P_w, \forall w\in W \tag{1}$$

$$\sum_{p\in P_w} x_{x,p} = r_w, \forall w\in W \tag{2}$$

Now we define a vector x_w that has components $x_{w,p}, p\in P_w$. Then the total flow pattern of the network can be described by a vector $X=(x_1,x_2,...,x_M)$. Obviously, X is subject to the constrain $X\in G$ where $G=G_1\times G_2\times ...G_w...\times G_m$ and G_w is simplex defined by Equ.(1) and (2). The total flow $F_{(u,v)}$ through a link $l_{(u,v)}$ in the network is related to X as follows,

$$F_{(u,v)} = \sum_{w\in W} \sum_{\substack{p\in P_w\\(u,v)\in L}} x_{w,p} \tag{3}$$

The cost function, which is assumed to be separable, can be expressed as

$$D=\sum_{(u,v)\in L} D_{(u,v)}(F_{(u,v)}) \tag{4}$$

As in most literature, including [7],[8], we assume that for each link $l_{(u,v)}\in L$, the function $D_{u,v}$, which is defined on $[0,\infty)$, is real valued, convex and twice continuously differentiable. The proposed optimal routing problem is to find a set of the path $\{x_{w,p}\}$ that minimizes the cost function D subjected to the constraints of Equ.(1) and (2). By expressing the total flows $F_{u,v}$ in terms of the path flows using (3), the problem can be formulated in the path flows space:

$$\begin{aligned} \text{minimize} \quad & \bar{D}(X) \\ \text{subject to} \quad & \sum_{p\in P_w} x_{w,p} = r_w, \forall w\in W \\ & x_{w,p}\geq 0, \forall p\in P_w, \forall w\in W, \end{aligned} \tag{5}$$

where X is the vector of path flow $x_{w,p}$. A frequently used method [7],[8] to solve Equ.(5) is used to convert the simplex constraints of the ORP into nonnegative constraints: In each iteration, for each commodity $w \in W$ a path $\bar{p}_w$ of minimum first derivative length. The path with minimum first derivative length is called the shortest path.

For each link $l_{(u,v)}$ and OD pair w , $D'_{(u,v)}$ is interpreted as the length of the pathflow for the link $l_{(u,v)}$ when the pathflow vector is x . In data communication routing and traffic assignment problem, the merit of the system is dependent on the values of marginal delay and travel time which are mainly the results from the selected algorithms used to solve the ORP problems. For all X and all $w \in W$, let

$$T_{(u,v)}(x,w) = D'_{(u,v)}(F_{(u,v)}), \tag{6}$$

$$T_{(u,v)}(x,w) \geq 0, \forall l_{(u,v)} \in L. \tag{7}$$

The length of the pathflow for $p \in P$ when the pathflow vector is X is defined by

$$L_p = \sum_{l_{(u,v)} \in P_w} T_{(u,v)}(x,w) \tag{8}$$

and the length of the pathflow for set P_w when the pathflow vector is X is defined by

$$L_w = \sum_{p \in P} L_p(x,w) \tag{9}$$

We then express the flow of $\bar{p}_w$ in terms of the other path flows while eliminating the equality constraints of Equ.(2):

$$x_{x,\bar{p}_w} \quad \approx \quad r_w - \sum_{\substack{p \in P_w \\ (u,v) \in L}} x_{w,p}, \quad w = 1,2,...,M \tag{10}$$

Plugging Equ.(10) into Equ.(5), we have the reduced dimension problem

$$\min_{\tilde{X} \in \tilde{G}} \tilde{D}(\tilde{X}) \tag{11}$$

where $\tilde{X} = (\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_w, ..., \tilde{X}_M)$ and $\tilde{X}_w$ is a vector with component $x_{w,p}, p \in P_w, w \in W, p \neq \bar{P}_w$; $\tilde{G}$ is defined to be $\tilde{G}_1 \times \tilde{G}_2 \times ...\tilde{G}_w... \times \tilde{G}_M$, with $\tilde{G}_w$ being a simplex defined by $x_{x,p} \geq 0, \forall p \in P_w, p \neq \bar{p}_w, \forall w \in W$. To solve the ORP represented by Equ.(11), the general algorithm takes the following iteration equation:

$$\tilde{X}_w(k+1) = \left[\tilde{X}_w(k) - \gamma^k H_w^{-1}(k)(L_w(k) - L_{\overline{w}}(k))\right]^+, \tag{12}$$

where $[\bullet]^+$ denotes the projection on $\tilde{G}_w$, $\tilde{X}_w(k)$ denotes the path flow vector of $x_{w,p}$, γ^k is a fixed relaxation parameter, $H_w^{-1}(k)$ denotes a positive and symmetric matrix of active pathflow which can be approximated by the use of the Hessian matrix of $\tilde{D}_w(X)$, $L_w(k)$ denotes the length of active path set P_w , the OD pair and $\bar{L}_w(k)$ denotes the length of the shortest path of the OD pair w . Also, the set W of OD pairs is divided into m commodities. Equ.(12) shows that general algorithm performs the computation for each commodity, w, sequentially. To parallelize the computation, [1] apply the following iteration equation using the global vector, $\tilde{X}$

$$\tilde{X}(k+1) = \left[\tilde{X}(k) - \gamma^k H^{-1}(k)(L(k) - \bar{L}(k))\right]^+, \tag{13}$$

where $[\bullet]^+$ denotes the projection on $\tilde{G}$. $H^{-1}(k)$ is a positive and symmetric matrix,its can be chosen to be an approximation of the Hessian matrix of $\tilde{D}(\tilde{X}(k))$. We used Equ.(13) to solve ORP problem by parallel.

At the beginning of the k-th iteration, a set of active path P_w^k consisting of at most (k-1) paths has been generated for the generic OD pair $w \in W$. The following calculation is executed sequentially from commodity no. 1 up to the last commodity, i.e. commodity number m. The OD pairs that have the same original can be found in same time by the shortest path algorithm. The GPM procedure can be given as follows:

Step 1: A shortest path that joins the original node for commodity with all other nodes in the same commodity is calculated. The length of path for each link $l_{(i,j)}$ used for this calculation is $T_{(i,j)}(x,w)$, where x is the current pathflow vector. These shortest paths are added to the corresponding list of active paths of each OD pair of the commodity if they are not already there, so the number of active paths for each OD pair of the commodity is at most k paths.

Step 2: Each OD pair w of the commodity is taken up sequentially. For each active path p of w, the length L_p as defined in Equ.(8) is calculated together with an additional number H_p^{-1} called the "step size". Both L_p and H_p^{-1} are calculated on the basis of the current total link flow vector. Let $\overline{p}$ be the shortest path obtained in Step 1 for the OD pair. The pathflows of all paths $p \neq \overline{p}$ are updated according to,

$$x_p(k+1) = \begin{cases} \max\{0, x_p(k) - \\ \quad \gamma^k H_p^{-1}(L_p(k) - L_{\overline{p}}(k))\}, \text{if } L_p > L_{\overline{p}} \\ \\ x_p, \text{Otherwise.} \end{cases} \tag{14}$$

The pathflow of the shortest path $\overline{p}$ is then adjusted by Equ.(10). In other words, an amount x_p or

$H_p^{-1}(L_p - L_{\overline{p}})$ is shifted from each non shortest path to the shortest path $\overline{p}$ which is smaller. The total link flows $F_{u,v}$ are adjusted to reflect the changes in x_p and $x_{\overline{p}}$. The GPM is adjusting path flow x_p until the condition in Equ.(15) is satisfied:

$$L_p - L_{\overline{p}} \geq \varepsilon, \forall p \text{ with } x_p > 0, \tag{15}$$

where ε is the error condition, $\varepsilon \approx 0$.

2.2 Step size approximation

In this section, the second algorithm for adjusting the pathflows is explained. To adjust all the pathflows in parallel, the consideration of the interaction of different commodities is necessary because the algorithm processes different commodities concurrently. Since the shortest paths of different commodities may share some communication links, parallel flow adjustments without considering the interactions among different commodities may cause the cost of the shared links to increase too much or even to be infinite and the process becomes unstable or divergent. This problem is recognized as the congestion problem. It should be pointed out that the congestion problem is not an issue for the general algorithms because they process different commodities one by one. This is a form of block Gauss-Seidel method that considers implicitly the interactions among different commodities. Iteration of Equ.(13) takes into account the interactions among different commodities. In the solution procedure of the general sequential ORP, only one processor with one global variable X is used. But in the solution method of the parallel ORP, m processors are used for m commodities and the global variable X is also expressed in m parts as $X = \{x_1, x_2, x_3, ..., x_m\}$. As the pathflows have been separated, each processor does not have to consider the pathflows belonging to other processors. This leads to the use of the Hessian matrix in the solution of the ORP as in Equ.(13). The equation of the Hessian matrix is as follows:

$$\begin{pmatrix} \boxed{\iddots} & & \\ & & \frac{\partial^2 \tilde{D}(\tilde{X}(k))}{\partial x_{pw1}\partial x_{pw2}} \\ & \boxed{\frac{\partial^2 \tilde{D}(\tilde{X}(k))}{\partial x_{pw}\partial x_{pw}}} & \\ & & \boxed{\iddots} \end{pmatrix} \tag{16}$$

The off-diagonal terms within the small boxes are corresponding to the interactions among different pathflows of the same commodity . The general algorithm [18] also ignores the off-diagonal terms within the small diagonal boxes since the interactions among the paths of the same commodity are not significant. The off-diagonal terms outside the small boxes reflect the interactions among the pathflows of different commodities. Even though the general algorithm does not explicitly use these off-diagonal terms, the block Gauss-Seidel iteration method applied by the general algorithm implicitly adapts the pathflows of different commodities since the pathflows of different commodities are adjusted sequentially during the outer iteration. However, since adjustment pathflow step done in parallel, the effect of the off-diagonal terms outside the small boxes needs to be considered. These off-diagonal terms can be calculated by:

$$\frac{\partial^2 \tilde{D}(\tilde{X}(k))}{\partial x_{p_{w1}}\partial x_{p_{w2}}} = \frac{\partial^2 \bar{D}(X(k))}{\partial x_{p_{w1}}\partial x_{p_{w2}}} - \frac{\partial^2 \bar{D}(X(k))}{\partial x_{\bar{p}_{w1}}\partial x_{p_{w2}}} \tag{17}$$

(Note that(17) can be derived by differentiation of following relation,

$$\frac{\partial \tilde{D}(\tilde{X}(k))}{\partial x_{p_{w1}}} = \frac{\partial \bar{D}(X(k))}{\partial x_{p_{w1}}} - \frac{\partial \bar{D}(X(k))}{\partial x_{\bar{p}_{w1}}}, \tag{18}$$

where $\bar{p}_{w_1}$ is the shortest path of commodity w_1 found at the k-th iteration. If the shortest path $\bar{p}_{w_2}$ of commodity w_2 share some common link, $\bar{L}$, with p_{w1}, from Equ.(17), it can be proved that:

$$\frac{\partial^2 \tilde{D}(\tilde{X}(k))}{\partial x_{p_{w1}}\partial x_{p_{w2}}} = \frac{\partial^2 \bar{D}(X(k))}{\partial x_{p_{w1}}\partial x_{p_{w2}}} + \sum_{(u,v)\in\bar{L}} D''_{(u,v)}(F(k)),$$

$$\sum_{(u,v)\in\bar{L}} D''_{(u,v)}(F(k)) > 0. \tag{19}$$

For every $x_{p_{w2}}$

$$\frac{\partial \tilde{D}(X(k))}{\partial x_{p_{w1}}\partial x_{p_{w2}}} > 0, p_{w2} \in P_{w2}. \tag{20}$$

If many commodities, W_{share}, have shortest path sharing a common link $l_{(u,v)_{share}}$ with commodity $\bar{p}_{w1}$, there will be a large number of such off-diagonal positive entries. It is not hard to show that ignoring these nonzero entries may cause too big flow adjustments calculated from (13) (a flow adjustment is define by $\tilde{x}_{w,p}(k) - \tilde{x}_{w,p}(k+1)$) corresponding to path $p_{w1}, p_{w1} \neq \bar{p}_{w1}$, and it also causes all such adjustments to be nonnegative for all $p_w, w \in W_{share}$,

$p_w \neq \bar{p}_w$. When Equ.(10) is executed, all of these adjustments will be added to link $l_{(u,v)_{share}}$ causing potential congestion problems.

There are two ways to solve the congestion problem: the first one is to choose a very small step size parameter, but this method resultes in more iterations. In general the bigger the network, the smaller the step size parameter needs to be chosen[8]. So, this method causes significantly longer computation time. The second method is properly adjust the associated diagonal term of $H(k)$ according to the congestion situation. It can effectively solve the congestion problems while still using the similar step size parameter (≈ 1) used in most general algorithms and thus, is much better. The second method is implemented by using a diagonal matrix

$$nl(k)=diag(nl_1(k)I_1,\cdot\cdot,nl_w(k)I_w,\cdot\cdot,nl_M(k)I_M), \qquad (21)$$

where $nl_w(k)I_w$ is again a diagonal matrix with $nl_w(k)$ being positive number (≥ 1) and I_w being a $(|P_w|-1) \times |P_w|-1)$ identity matrix. Here, $nl_w(k)$ is called compensation matrix that is used to compensate the effect of ignoring the off-diagonal terms of the Hessian matrix.

Let $H(k) = nl(k) \times DH$, where DH is a diagonal matrix with the same dimension of the Hessian matrix of $\tilde{D}(k)$ but only keeps the diagonal terms of the Hessian matrix. $nl_w(k)$ is found as follows:

For commodity w, at kth iteration, after all shortest paths are found, all the links of the shortest path $\bar{p}_w$ are counted for the number of all sharing shortest path and $nl_w(k)$ is the largest of all these numbers; the associated link(s) are named as the critical link(s) of commodity w at kth iteration. Accordingly, the number of shortest paths sharing the critical link of w is $nl_w(k)$.

A simple example is illustrated in Fig. 1. In This case, commodity 1,2 and 3 have shortest path p_1, p_2 and p_3 found, respectively. For commodity 1, the critical links,(5,6) and $nl_1(k)$ is 3 ;commodity 2 the critical link is (5,6)and (6,ancestor), and $nl_2(k)$ is 3, and so on. With our scheme, the flow adjustment at the current iteration for every path of commodity 1 is $1/nl_1(k) = (1/3)$ of the original flow adjustment and same for commodities 2 and 3. As a result, the total flow adjustment at critical link(s) is less than or equal to $1/nl_w(k)$ of the original total flow adjustment.

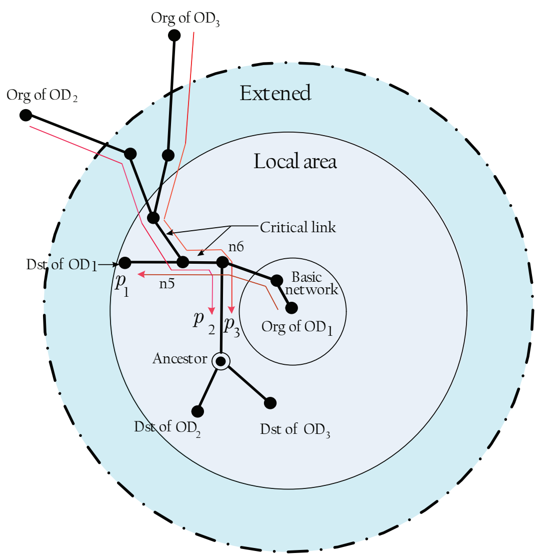


Figure 1: Overlapped parts

3 Definitions Assumptions for Optimal routing algorithm

3.1 Definitions

Definition 1 *A path p is a finite sequence of link $p = (l_{(u_1,u_2)}, l_{(u_2,u_3)}, ..., l_{(u_{k-1},u_k)})$. The path of derivative length L_p is define to be $L_p = T_{(u_1,u_2)} + T_{(u_2,u_3)} + ... + T_{(u_{k-1},u_k)}$ where $T_{(u_{i-1},u_i)}$ is the result of the first derivative of the objective function $D_{(u_{i-1},u_i)}$ with respect to the flow on link $l_{(u_{i-1},u_i)}$.*

The hop length $H(P)$ of a path is defined as the number of link in the path. The minimum hop distance (MHD) between two nodes, u, and v is represented by $MHD(u,v)$ and is defined by $MHD(u,v) = \min_p\{H(P)$ for all P starting at u and ending at $v\}$.

Definition 2 *Diameter Φ of a network G, where $G = (V, L)$, is defined to be $\Phi = \max_{u,v \in V} MHD(u,v)$*

Definition 3 *Time complexity of an algorithm or a procedure is defined to be the sum of the computation time complexity and communication time complexity that are needed to execute the algorithm or the procedure in distributed computation environment*

In the following section, the definitions related to the hierarchically structure network topology are described. For an extremely large data network, the topology often possesses some hierarchical structure which can be characterized conceptually with a balance-tree hierarchy (BTH) [3],[16].

A complete tree means a rooted tree in which all leaves are of the same depth. A network can be represented by a BTH with the same number of leaves as the number of nodes in that network. All the levels of a BTH are numbered in such a way that the root has the highest level number and the leaves have the level number of 1.

In the following section, the hierarchically structure network topology is defined. Firstly, the concept of network super-set S is introduced. Each element s in S is a bounded node set of network and the subnetwork composed of the nodes in each node set should be connected. A special case is that a node set includes only one node. Such a set is called a single node set. The union of single node set in a network super-set must include all the network nodes.

Definition 4 *A connected network is said to be hierarchically structure if there exists a mapping, $P : T \rightarrow S$, from a BTH, T, to a super-set, S, of the network, such that:*

1. *P map every leaf in T to every single node set in S;*
2. *If $s^l_{m1}, s^l_{m2}, ..., s^l_{ml}$ are the images of the elements $t^l_{m1}, t^l_{m2}, ..., t^l_{ml}$,which are the entire nodes at lth level in the BTH, the subnetwork composed of all the nodes of $s^l_{m1} \bigcup s^l_{m2} ... \bigcup s^l_{ml}$ must be connected;*
3. *If $t^{l-1}_{n1}, t^{l-1}_{n2}, ..., t^{l-1}_{nj}$ are the complete child nodes of t^l_j and $t^{l-1}_{n1}, t^{l-1}_{n2}, ..., t^{l-1}_{nj}$ and s^l_j are the images of $t^{l-1}_{n1}, t^{l-1}_{n2}, ..., t^{l-1}_{nj}$ and t^l_j , respectively, then $s^l_j \subset s^{l-1}_{n2} \bigcup s^{l-1}_{n2} \bigcup ... \bigcup s^{l-1}_{nj}$ and the subnetwork composed of all the nodes in $s^{l-1}_{n2} \bigcup s^{l-1}_{n2} \bigcup ... \bigcup s^{l-1}_{nj}$ must be connected.*

Fig. 2 shows an example of mapping from a BTH to a network. We will refer to the node in a node set, s, as a group. Since we have created a mapping between a BTH and a network, we can adopt all the relational terms of BTH, such as, parent and child in describing the relation ship of subnetworks or groups.

Definition 5 *The subnetwork composed of all the groups (and the associated links) having a common parent group (we also refer to a parent group as a gateway group) is defined as basic subnetwork. The nodes in the gateway groups are call gateways.*

Definition 6 *Let V_l be the union of all the node set that are the images of all the lth level nodes of the associated BTH and let L_l be all the links associated with V_l. Similarly, Let V_{l+1} denote the union of all node sets that are the images of all the $l + 1$th level nodes of the BTH and let L_{l+1} denote the links associated with V_{l+1} Then, the layer of a network is defined to be $G_l = (V_l, L_l - L_{l+1})$.*

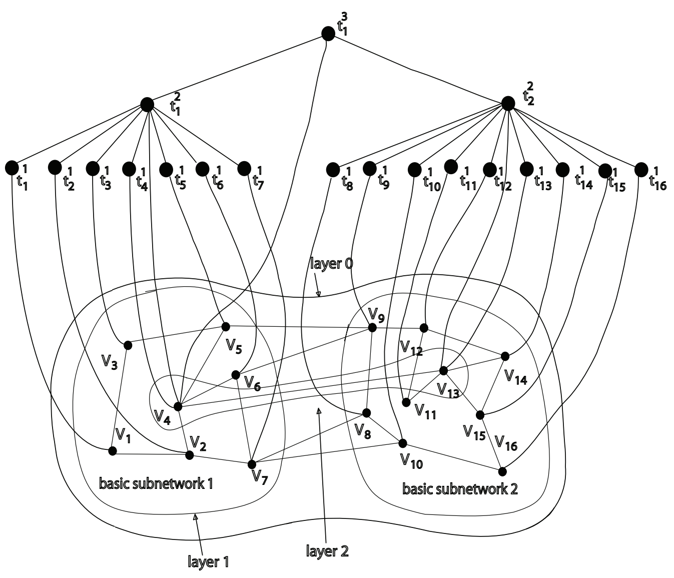

t^3_1
t^2_1
t^2_2
t^1_1 t^1_2 t^1_3 t^1_4 t^1_5 t^1_6 t^1_7 t^1_8 t^1_9 t^1_{10} t^1_{11} t^1_{12} t^1_{13} t^1_{14} t^1_{15} t^1_{16}
layer 0
V_1 V_2 V_3 V_4 V_5 V_6 V_7 V_8 V_9 V_{10} V_{11} V_{12} V_{13} V_{14} V_{15} V_{16}
basic subnetwork 1
basic subnetwork 2
layer 1
layer 2

Figure 2: An example of a mapping from a BTH ($T = \{t^1_1, t^1_2, ..., t^1_{16}, t^2_1, t^2_2, t^3_1\}$) to a super-set of a network ($S = \{s_1, s_2, ..., s_{16}\} = \{\{v_1\}, \{v_2\}, ..., \{v_{16}\}\}$)

3.2 Network Assumptions

In this section, the hypotheses of the network for our algorithm are as below:

Assumption 1 *The network is hierarchically structured*

Assumption 2 *Each node has a unique ID taken from $\{1, 2, ...n\}$.*

Assumption 3 *For a network $G = (V, L)$, if a link $l_{(u,v)} \in L$, we also have $l_{(v,u)} \in L$. This means if node u has a link to node v, node v also has a link to node u.*

Assumption 4 *The maximum ratio of the first derivatives of the link cost functions*

$$\max_{(u,v),(k,l)\in L} D'_{(u,v)}/D'_{(k,l)}$$

is bounded. We let

$$B = \max_{(u,v),(k,l)\in L} D'_{(u,v)}/D'_{(k,l)}$$

And call B as the link upper bound.

Assumption 5 *Each gateway group of the network has at least one processor. For convenience, we assume that each gateway group has exactly one processor. More general cases that each gateway group has more than on processor can be easily extended.*

Assumption 6 *The maximum degree of the network is bounded by a constant. Here, the degree is the number of outgoing links from a node.*

Assumption 7 *Finite (or bounded) step1s of computations or transmissions associated with $O(1)$ length of data can be completed in $O(1)$ time units.*

Assumption 2, 3 and 5-7 are quite standard for most data networks. Assumption 4 is also reasonable since if $D'_{(u,v)}/D'_{(k,l)}$ is unbounded, based on the general premise of the link cost function, it implies some link cost goes to infinity (since $D'_{(k,l)}$ cannot be zero.) The network is then in an abnormal condition, which is beyond the scope of the thesis. However, a smaller bound B will result in better performance of general hierarchically data structure ORP algorithm.

4 Optimal routing algorithm

4.1 Reverse shortest path

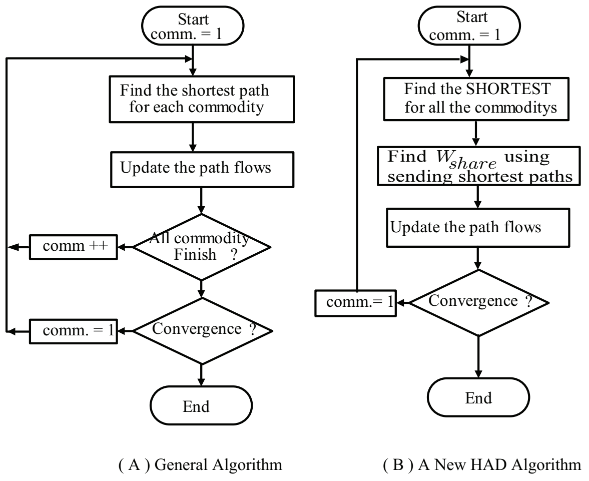
In the process of finding the solution for the ORP, there are two important steps of calculation, they are, a) to find for the shortest path in each iteration , and b) to adjust the path flow. We can compare the conventional algorithms to our algorithm as shown in Fig. 3.

General algorithm for ORP [2],[13],[14] is shown in Fig. 3.(A), the new HAD algorithm is shown in Fig. 3.(B) and in Fig. 3.(C) is our proposed algorithm.

The demand $r_{(u,v)}$ is to be distributed to smaller parts such as $r_{(u,v)} = r_{(u,j)} = r_{(j,k)} = r_{(k,v)}$ where j is ancestor and k is gateway. As the consequence of this distribution, the demand outside the local area is taken into considerations in each iteration of the shortest path calculation. This leads us to find the reverse shortest path to locate the paths between commodities outside the local area and the gateway or ancestor inside the local area which is a direct proportion to the number of destination nodes outside the local area as shown in Fig. 4(A). This enables us to find the shortest path to the common destination via only a single calculation for the reverse shortest path within the time of $O\log(n)$ as shown in Fig. 4(B).

The process to find the reverse shortest path is almost similar to the process for the shortest path as shown in algorithm:

```
1: Initial: Let $N = \{dst\}$, for each $j \notin N$ set $C(j) = d_{(j,i)}$
2: repeat
3:     Choose $j \notin N$ with $\min C(j)$
4:     Set $N = N \cup \{j\}$
5:     Update $C(j)$: For all $j \in N$
       set $C(j) = \min(C(j), C(i) + d_{(j,i)})$
```



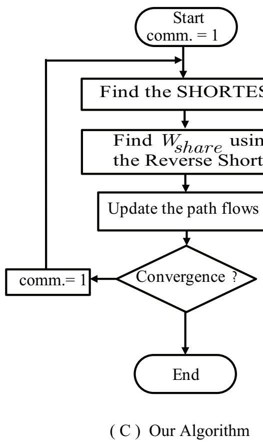


Figure 3: A Comparison of general algorithm procedure, A new HAD algorithm and our algorithm procedure.

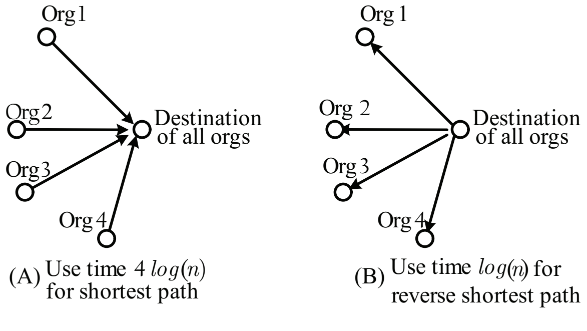

Org1
Org2
Org3
Org4
Destination of all orgs
(A) Use time $4\,log(n)$ for shortest path
Org 1
Org 2
Org 3
Org 4
Destination of all orgs
(B) Use time $log(n)$ for reverse shortest path

Figure 4: complexity revers shortest

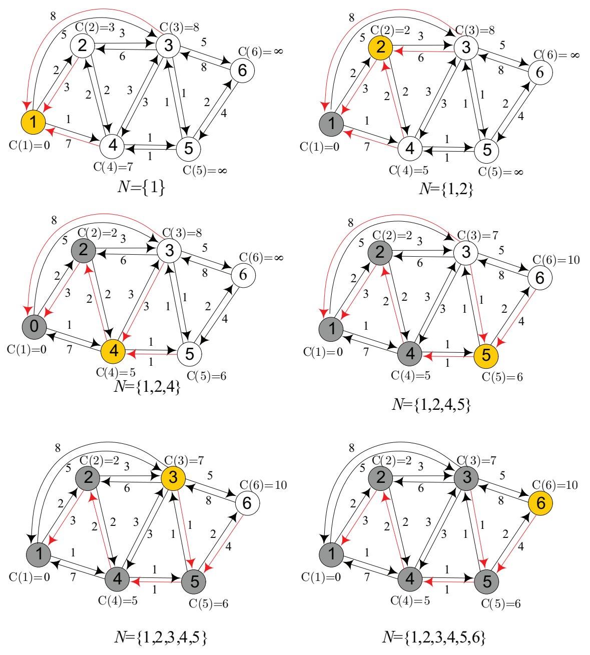


Figure 5: The Reverse Shortest Path Algorithm

```
6: until (all starting node $\in N$)
```

At the start of the process, let $N = \{dst\}$ denote the set of nodes in the network. The shortest paths are then reversely calculated from the destination nodes back to the original nodes as shown in Fig. 5.

Fig. 5 Example results from the application of the Reverse Shortest Path algorithm.

Starting from the destination node 1, the method finds the end nodes of the incoming links and then calculates the cost of node 2, $C(2) = 3$,the cost of node 3, $C(3) = 8$ and the cost of node 4, $C(4) = 7$. Then the node with $C(j)$ is selected, that is node 2. Let $i = 2$ and place node 2 into the set of scan nodes, $N = \{1, 2\}$. Starting from node 2, one updates the cost of node 4, $C(4) = 5$. The operating process of the Reverse Shortest Path algorithm stops when all the considered nodes has been placed in N.

To understand the Reverse Shortest Path algorithm, consider the following claims.

Proposition 7 *Proposition 1: At the beginning of iteration,*

1. $C(i) \leq C(j)$ *for all* $i \in N$ *and* $j \notin N$
2. $C(j)$ *is, for all* j *, the shortest path from dst to* j *using paths with all nodes in* N. $C(j)$ *is, for all* j *, the shortest path from dst to* j *using paths with all nodes in* N

If this proposition can be proved, then it can be assumed that, when contains every node in then all are shortest paths by the second part of this proposition. Therefore proving the above proposition is equivalent to prove the Reverse Shortest Path algorithm.

Proof: The proposition is trivially true at the first step since N consists only of the single destination node (node $dest$) and $C(dst) = 0$ for $j = dst$,$d_{(j,i)} \geq 0$ for nodes reachable directly from node dst, and ∞ otherwise. The first condition is simply shown to be satisfied since it is preserved by the formula,

$$C(j) = min\left[C(j), C(i) + d_{(j,i)}\right] \qquad (22)$$

which is applied to all $j \notin N$ when node i is added to set N. We show the second condition by induction. We have established already that it is true at the very start of the algorithm. Next, let assume that it is true for the beginning of some iteration of the algorithm. It will be shown that the proposition must be true at the beginning of next iteration. Let node i be the node added to set N and let C_k be the label of node k at the beginning of the step. The second condition must, therefore, hold for node $j = i$ (the new added node) by the hypothesis. It must also hold for all nodes $j \in N$ by part one of the proposition which is already proven. It remains to prove that the second condition of proposition is met for $j \notin N \cup \{i\}$.

Consider a path from dst to j which is the shortest amongst those with all nodes except j in $N \cup \{i\}$ and let C'_j be the corresponding shortest cost. Such a path must contain a path from dst to some node $r \in N \cup \{i\}$ and $l_{(r,j)}$. We have already established that the length of the path form dst to r must be C_r and therefore:

$$\begin{aligned} C'(j) &= \min\left[C(r), C(i) + d_{(j,r)}\right], \\ &= \min\left[\min_{r \in N}[C(r) + d_{(r,j)}], C(i) + d_{(j,i)}\right]. \end{aligned}$$

However, since $C(j) = \min[C(r), C(i) + d_{(j,r)}]$, therefore,

$$C'(j) = \min\left[C(j), C(i) + d_{(j,i)}\right]. \qquad (23)$$

which is exactly what is set by the third step of the algorithm. Thus, after any iteration of the algorithm, the second part of the proposition is true if it is true at the beginning of the iteration. Thus the proof by induction is completed.

The comparison between the Reverse Shortest Path algorithm and other shortest path algorithms has been carried on by comparing the operation results from using each algorithm to find the shortest paths from the source nodes 3, 5, 6 to the destination node 1 in Fig. 6(A,B,C). For the operations using the Dijkstra's algorithm or the Bellman-Ford's algorithm, the

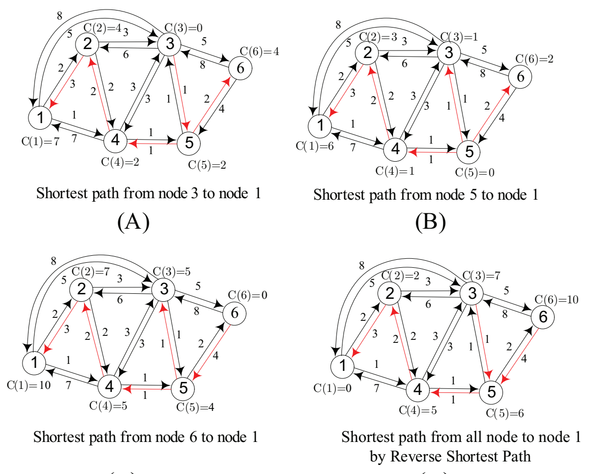

C(2)=4
C(3)=0
C(6)=4
C(1)=7
C(4)=2
C(5)=2
Shortest path from node 3 to node 1
(A)
C(2)=3
C(3)=1
C(6)=2
C(1)=6
C(4)=1
C(5)=0
Shortest path from node 5 to node 1
(B)
C(2)=7
C(3)=5
C(6)=0
C(1)=10
C(4)=5
C(5)=4
Shortest path from node 6 to node 1
(C)
C(2)=2
C(3)=7
C(6)=10
C(1)=0
C(4)=5
C(5)=6
Shortest path from all node to node 1 by Reverse Shortest Path
(D)

Figure 6: comparison of general shortest path and Reverse Shortest Path

shortest path calculation has to be done three times to find the shortest path from 3 to 1, from 5 to 1, and from 6 to 1, with the consumed time of $|V||L|$ when V is set of node and L is the set of links $l_{(i,j)}$. But, for the operation using the Reverse Shortest Path algorithm, the task can be accomplished within only one calculation with the time complexity of $O(\log(n))$. The results from Reverse Shortest Path algorithm in Fig. 6(D).

In [1], a scheme of the SHORTEST_PATH algorithm was proposed by carrying on the shortest path calculations in parallel along the local area. Each local area has $m(\ell, i)$ commodities with ℓ the layer of subnetwork, i the sequence of b_ℓ local area. The parallel shortest path was carried on local area with the number of commodities of

$$m = \max_{1\ell \leq L; 1 \leq i \leq b_i} m(\ell, i) \qquad (24)$$

In this research, we proposed Reverse Shortest Path algorithm in order to find the shortest path from all destinations to origin gateway of local area, the number of gateway per local area is 1, time spent in calculation of the shortest path becomes $(O\log(n))$ as shown in the flowchart in Fig. 3(C).

In our algorithm, the reactions from demands of commodities outside the local area for using in reverse shortest path procedure, this algorithm decrees interchange message among processor.

```
Step 1. Initialization: find a feasible solution for the
        routing problem

Step 2. while (termination condition is not satisfied)

  Step 2.1 call SHORTESTPATH

  Step 2.2 call rev_shortest by origin is ancestor/gateway,
           destination is origin from reactions, then find $nl_w(k)$.

  Step 2.3 Adjust path flow for each path.

  Step 2.4 check the termination condition

  End of while.
```

In the initialization of step 2, a feasible solution can be obtained from the SHORTEST_PATH with distributed flow demand through the calculation of the minimum hop distance for each commodity together with the resulting commodities from the flow demand distribution. The path flow for each shortest path should not exceed the link capacity. In case of larger path flow, the flow demand has to be divided and then run the SHORTES_TPATH with distribute demand to find the shortest path for flow demand. This step can be finished within $O((m)\log(n))$ time units. Substep 2.1 takes the time of $O(m)\log(n)$ Substep 2.2 takes the time of $O(\log(n))$ time. Substep 2.3 takes the time of $O(m\log(n))$ units. Step 2.4, the terminal condition is tested for every process takes the time of $O(log(n))$.

5 Test results

The test run on the proposed algorithm has been performed by simulating the network using 20 Personal computer with 2.4 GHz processor 256 MB SDRAM . Each PC was programmed to solve the ORP for each local area with the network structure as shown in Fig. 7 with 200 nodes structured into 3 layers. The simulation was carried on using the *Monte Carlo* simulation scheme . Test results ware then compared to the test-run results for the New HAD algorithm [1] on the same network simulation and the same link of 10K packets/sec with the packet average of 64 bytes. K packets/sec giving the cost function for the link$l_{(u,v)}$ as in Equ.(25)

$$D_{(u,v)}(F_{(u,v)}) = \frac{F_{(u,v)}}{C_{(u,v)} - F_{(u,v)}} \qquad (25)$$

The queuing model used in the test is the $M/M/1$ queuing model with $C_{(u,v)}$ and $F_{(u,v)}$ as the link capacity and link flow of the same unit of link from node u to node v. The test on the proposed algorithm was carried on using the *Monte Carlo* pattern as concluded [1] in Table 1 and Table 2.

The 200 nodes in the test networks are divided into 20 clusters of basic networks. For each basic network, one processor is programmed to solve the ORP for the commodities in each basic network The whole test system needs 20 processors . Each basic network has the ancestor to work as the gateway of layer 0 ,

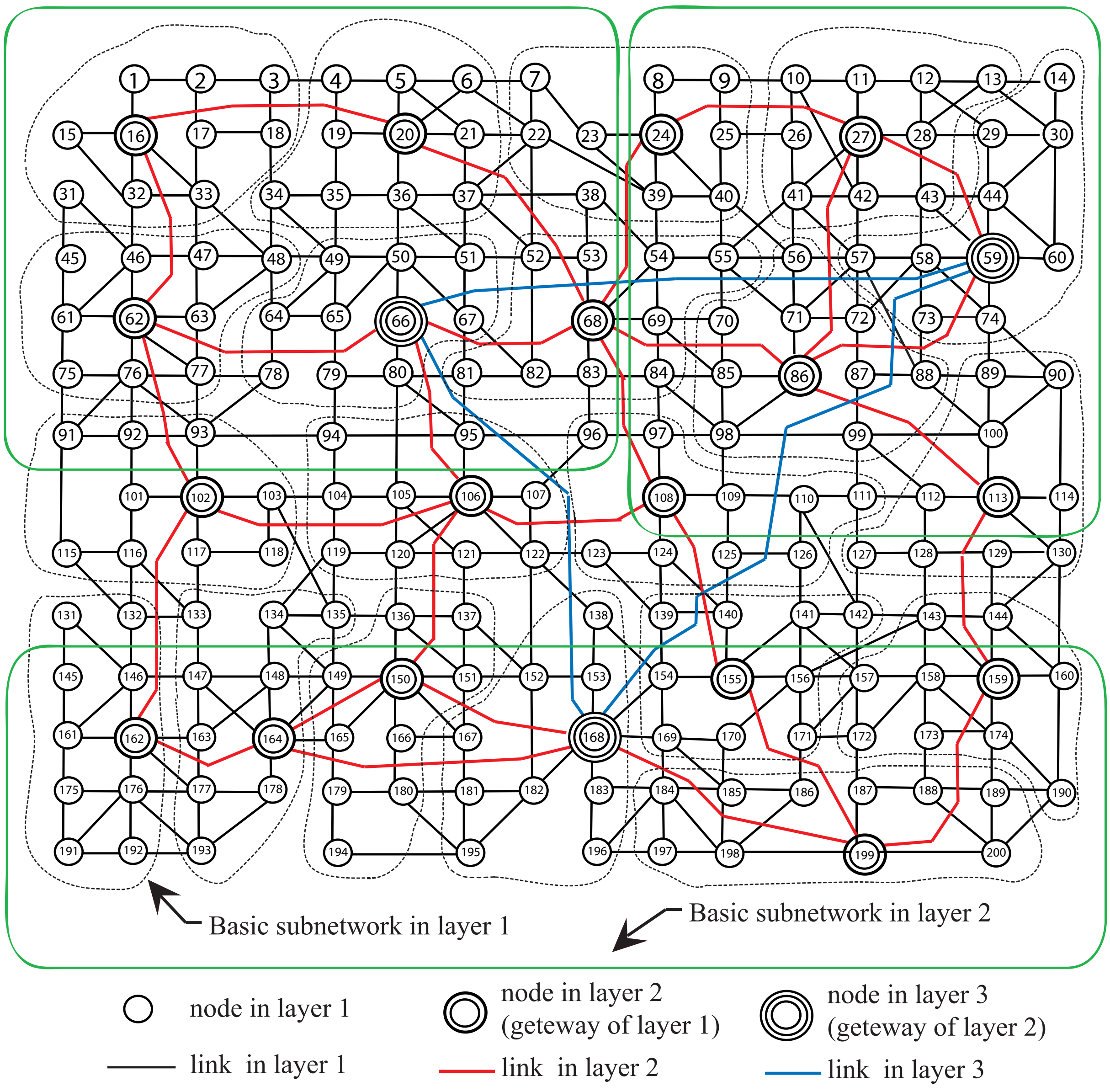


Figure 7: Network example

the basic networks with the same gateway are then grouped together called the local area so that each local area has one gateway. The test network has three local areas and three gateways, where all the gateways are in layer 2 with the fixed relaxation parameter of 1. Test algorithm has been developed using C on the operating system Linux with the fixed relaxation parameter of $\gamma^k = \gamma_w^k = 1$ and the error condition of $\varepsilon = 0.1$. Test results as shown in Table 3 and Table 4 show that our proposed algorithm give a better speed than the compared algorithm. Our algorithm also helps to reduce the communication time between processors.

6 Conclusion

In this paper, we present a new algorithm to reduce the steps in New HAD optimal routing algorithm for large networks by the distribution of demands and solve the shortest path problems with the Reversed Shortest Path method which enables the ORP processes for large hierarchically structured networks calculations of a large Hessian matrix. This proposed algorithm gives a better speed than other algorithms that use the multi-commodity hierarchically structured networks technique. The proposed algorithm has been tested with an example hierarchically structured network with 200 nodes.

Pattern	#comm. gen.	Dist of # of comm. against diff MHD										
		t_{mhd_i}	1	2	3	4	5	6	7	8	9	>10
1	101		30	15	14	10	7	6	7	5	5	2
2	99	#of	28	15	13	10	8	7	5	6	3	4
3	100	com	31	14	13	11	9	9	5	4	3	1
4	102	*OD*	31	20	10	12	10	5	4	5	4	3
5	104		32	18	14	10	10	10	4	3	2	1

Table 1: The Statistics of five patterns of the commodities generated by Scheme 1.

Pattern	# comm. gen.	Dist of # of comm. against diff MHD										
		t_{mhd_i}	1	2	3	4	5	6	7	8	9	>10
6	101		8	9	11	16	15	11	11	7	7	7
7	99	#of	7	11	12	12	12	8	11	12	8	7
8	100	com	5	12	15	14	15	7	7	11	12	9
9	102	*OD*	5	6	13	12	12	10	10	11	11	5
10	104		7	15	18	14	10	15	5	5	9	5

Table 2: The Statistics of five patterns of the commodities generated by Scheme 2.

New HAD algorithm		Our algorithm		Speedup	Cost
Parallel Runtime	Cost	Parallel Runtime	Cost	New HAD vs OUR	error %
9.50	73.2	8.01	73.2	1.19	0
8.30	74.5	6.80	74.5	1.22	0
8.61	73.2	7.40	73.2	1.16	0
10.82	79.7	8.8	79.7	1.30	0
10.24	72.5	8.67	72.5	1.18	0

Table 3: The test results of pattern 1∼5.

New HAD algorithm		Our algorithm		Speedup	Cost
Parallel Runtime	Cost	Parallel Runtime	Cost	New HAD vs OUR	error %
19.1	83.2	15.60	83.2	1.22	0
19.5	82.2	16.38	82.2	1.19	0
20.15	84.5	17.63	84.5	1.14	0
20.0	80.7	17.0	80.7	1.18	0
24.6	80.0	20.17	80	1.22	0

Table 4: The test results of pattern 6∼10.

References:

[1] Garng M. Huang and Shan Zhu, A new HAD algorithm for optimal routing of hierarchically structured data network, *IEEE transactions on parallel and distributed systems.*, 7(9), September 1996.

[2] W.K. Tsai, G.M. Huang,J.K. Antonio,and W.T. Tsai. Aggregation/Disaggregation Algorithm for Optimal Routing in Data Network, *Proc Automat. Contr. Conf.*, Atlanta, Ga., June 1988.

[3] J.K. Antonio,G.M Huang and W.K.Tsai, A Fast Distributed Shortest Path Algorithm for a Class of Hierarchically Clustered Data Networks, *IEEE Transaction on Computer.*, Vol.41, No.6, pp.710-712, June 1992.

[4] R.Gallager, A Minimum Delay Routing Algorithm Using Distributed Computation, *IEEE Trans. Communication.*, Vol.25, pp.73-85, 1977.

[5] G.M Hung and W. Hsieh, Exact Convergence of a Parallel Textured Algorithm for Data Network Optimal Routing, *IEEE Transaction on Parallel and Distributed Systems.*, Vol.6, no.11, pp.1-15, Nov.1995.

[6] G.M. Huang and W. Hsieh, A parallel textured algorithm for optimal routing in data networks, *Proc IEEE Global Com 91.*, Ariz., December 1991.

[7] J.Tsitsiklis and D.Bertsekas, Distributed Asynchronous Optimal Routing in Data Networks, *IEEE Transaction on Automatic Control.*, Vol.31, no.4, pp.325-332, 1986.

[8] D.Bertsekas and J. Tsitsiklis, *Parallel and distributed computation,Numerical Methods.*Englewood Cliffs, N.J., Prentice Hall., 1989.

[9] D.Bertsekas, E.M.Gafni and R. Gallager, Second Derivative Algorithm for Minimum Delay Distributed Routing in Network, *IEEE Trans. Communication.*, Vol.32, No.8, pp.911-919, 1984.

[10] D.Bertsekas, E.M.Gafni, Projected Newton Methods and Optimization of Multi-commodity Flows, *IEEE Trans. Automatic Control.*, Vol.28, No.12, pp.1,090-1,096, 1983.

[11] D.P. Dbetsekas, Projected Newton Methods for Optimization Problems with Simple Constraints, *SIAMJ. Control and Optimization.*, Vol.20, pp.221-246, 1982.

[12] J.K. Antonio,W.K. Tsai and G.M. Huang, Time Complexity of Path Formulated Optimal Routing Algorithm, *IEEE Trans.Automatic Control.*, Vol.39, No.2, 1994.

[13] S.W.Park and W.K. Tsai, Distributed Hierarchical Optimal Routing using Aggregation/Disaggregation and Decomposition/Composition Techniques, *Proc Int'l*

Parallel Processing Symp., Anaheim, Calif., 1991.

[14] S.W.Park, Routing Algorithm in Communications Network, PhD dissertation, Dept. of Electrical and Computer Engineering, Univ. of California, Irvine, 1991.

[15] G.M. Huang and S. Zhu, A New Distributed Shortest Path Algorithm for Hierarchically Clustered Data Networks, *Proc IEEE am. Control Conf.*, Seattle, Wash., June, 1995.

[16] A.V.Aho,J.E.Hoptcrft and J.D. Ullman, *The Design and Analysis of Computer Algorithms.*Reading,Mass.:Adison-Wesley, 1974.

[17] S.W.Park, Queueing Systems: Vol.II, Computer Applications.,New York: John Wiley & Sons, 1976.

[18] R.Cheng, and M.Gen, A priority base encoding and shortest path problem, *Ashikaga Institute of Technology.*, 1998.

[19] MOUSTAPHA DIABY, Traveling Sale Man Problem : A Linear Programming Formulation, *WSEAS TRANSACTION ON MATHEMATICS.*, Vol.6, Issue 6, 6 June 2007.

[20] F. GONZALEZ, I. DE MIGUEL, P. FERNANDEZ, J.C. AGUADO, R.M. LORENZO, E.J. ABRIL, M. LOPEZ, Iterative Linear Programming Formulation for Routing and Wavelength Assignment in Optical Networks, *WSEAS Signal Processing, Communications and Computer Science.*, pp. 57-62, 2000.

[21] KAIRAT JAROENRAT, CHOM KIMPAN, PISIT CHARNKEITKONG AStudy on the Flow Assignment Efficiency of MENTOR Algorithm, *Proceedings of the 10th WSEAS International Conference on COMMUNICATIONS.*,Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp419-424).