Journal of the Korean
OR/MS Society
Vol. 11, No. 2, Oct. 1986

—Research Paper—

A Heuristic for Non-bifurcated Routing in Communication Networks

Suk-gwon Chang*

ABSTRACT

This paper considers the routing problem in session-based communication networks, particularly where the pattern of message transmission is characterized by non-bifurcated traffic flows. Due to the discreteness property of its solution space, the problem is formulated as normalized multicommodity network flows with integer restrictions and an effective penalty system is suggested to solve it. To show its effectiveness, a simple example and a preliminary computational experience are provided. Finally, numerous application areas of the algorithm are suggested.

1. Introduction

In a message (or packet) switched communication network, non-bifurcated flows characterize the pattern of message transmission of static and quasistatic routing, in which routes are defined at system generation or at session initiation for each pair of communicating nodes. With this type of routing, availability of sufficient bandwidth as well as delay plays an important role in routing decisions. The non-bifurcated routing is usually formulated as a nonlinear integer programming, for which analytical techniques are not as well developed as those for non-integer programming.

The purpose of this paper is to suggest a mathematical tool for handling directly the routing or route selection problem formulated as a normalized multicommodity network flow problem with additional integer restrictions. Although the suggested algorithm lacks in its theoretical exactness due to the innate intractability of the routing problem involved, its simplicity and effectiveness in both design and implementation point of view compensate for this incompleteness.

Since the non-bifurcated routing was first identified by Fratta, et al. [3] in their optimal flow assignment problem formulated as a nonlinear cost multicommodity network flow problem, there have been some researches recently on this non-bifurcated routing such as Gavish [5] and Tcha [9]. In their works, non-bifurcation of flows was formulated explicitly as multiple choice constraints to model the operating environment of IBM's SNA networks which use a fixed routing scheme.

*Dept. of Business Administration, Hanyang University.

The heuristic algorithm suggested in this paper may also be thought of as a variant of their algorithms, but quite differs from them in its simplicity and algorithmic flexibility ; say, a variety of performance criteria can be included in our algorithm with an appropriate variable transformation. We assume first an SNA-like networking environment which prespecifies a set of communication paths for each node pair where sessions will be created. This assumption is introduced mainly for exposition brevity and is not essential for the application of our algorithm. It will be shown later that the algorithm performs well for a TYMNET-like networking environment where a route has to be selected from the network itself.

This paper is structured as follows : Section 2 introduces a multiple choice type of formulation which is the normalized arc-chain formulation of multicommodity network flows. Then it shows that a variety of performance criteria can be included in the same framework. Section 3 presents a prototype of our routing algorithm and discusses some possible algorithmic variations. Section 4 performs a network realization of the prototype algorithm introduced in Section 3. Section 5 presents a simple example and a preliminary computational experience obtained with a FORTRAN code of the algorithm. Finally, Section 6 concludes and suggests some application areas.

2. Problem Formulation

Consider a communication network modeled by a directed graph $G=(N, L)$, $N=\{1, 2, \dots, N\}$ is the set of N nodes and $L=\{1, 2, \dots, L\}$ is the set of L distinct directed links. Assume that there are K ordered pair of nodes where sessions will be created. Denote the corresponding set of ordered pairs by $K=\{1, 2, \dots, K\}$. Note that each of N, L and K is used to denote the set as well as its cardinality. Then to each ordered node pair corresponds a set of paths S_k for $k \in K$ preselected by the system operator. These S_k's $(k=1, 2, \dots, K)$, having the property $S_i \cap S_j = \emptyset$ for $i \neq j$, constitute the entire path set S. Each link i for $i \in L$ is limited in its data transmission capability with maximum rate C_i (kbps). These will be refered to as the capacities of the links.

Let, at a given moment, a set of incoming sessions be presented to the network while assuming, only for simplicity, there do not exist any sessions previously connected. We assume also that the traffic of these incoming sessions are previously estimated and denote them by Δ_k(messages/sec) for each pair k in K. Then under the well-known standard set of nice assumptions of an open queueing network (see [7]), the problem of allocating the incoming sessions to the appropriate routes is modeled, adopting the average message delay as our performance criterion, as follows :

$$\text{Minimize} \quad g(\rho) = 1/\gamma \sum_{i \in L} \rho_i/(1-\rho_i) \tag{1}$$

$$\text{subject to} \quad Ax = \rho \tag{2a}$$

$$x \in X \tag{2b}$$

$$0 \leqq \rho_i \leqq 1, \; i \in L \tag{2c}$$

where γ(messages/sec) is the total traffic entering the network, ρ is an $L \times 1$ vector whose $i-th$ element is the resultant link utilization factor ρ_i, $1/\mu$ is the average message length, A is an $L \times |S|$ matrix whose $(i, j)-th$ element is

$$a_{ij} = \begin{cases} \Delta_k/\mu C_i & \text{if path } j \in S_k \text{ traverses link } i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

(Note that A and its corresponding link (arc)–path incidence matrix have the relationship that the locations of non-zero entries in both matrices are equal each other when all the Δ_k's are positive) and

$$X = \{ x \mid \sum_{j \in S_k} x_j = 1,\ k \in K,\ x_j = 0 \text{ or } 1,\ \ j \in S \} \tag{4}$$

$g(\rho)$ of (1) is a scalar function whose value represents the average message delay. Though we began with this system-oriented performance criterion, all the constraints of (2) remain effective when some other criteria such as the user-oriented message delay (refer to [8] for its definition), maximum link delay and maximum link utilization factor are used instead.

The constraints ($2a$) and ($2c$) require that every link utilization factor resulting from the set of established paths be not greater than one. Note that a_{ij} represents the increment of the link utilization factor of link i when path j is selected. The constraints ($2b$) and (4) are so-called multiple choice constraints which require that only one path be selected from each set of candidate paths S_k for $k \in K$.

Let us introduce the vector s whose $i-th$ component is defined by

$$s_i = 1 - \rho_i \tag{5}$$

which represents the residual utilization factor of link i. Then the problem can be expressed as the following more comfortable form.

$$\text{Minimize } \ f(s) = 1/\gamma \sum_{i \in L} (1 - s_i)/s_i \tag{6}$$

$$\text{subject to} \quad Ax + s = 1 \tag{7a}$$

$$x \in X \tag{7b}$$

$$s \geqq 0 \tag{7c}$$

which will be refered to as the model P.

As mentioned previously, if the criterion of minimizing the maximum link delay is adopted,

$$f(s) = \max_{i \in L} \quad 1/\mu C_i s_i \tag{8}$$

and for the one of minimizing the maximum link utilization factor

$$f(s) = -\min_{i \in L} s_i \tag{9}$$

The last two forms of $f(s)$ draw our special attention in that the corresponding model (8) or (9) with (7) can be transformed into a linear one after trivial manipulations (see [9]). Then the resulting 0-1 linear programming with multiple choice constraints, often refered to as multiple choice programming, can be solved by an efficient heuristic recently developed by Chang and Tcha [1].

We should note here that the solution space defined by the constraints (2) is discrete in nature, preventing application of the well-known Kuhn-Tucker optimality condition which holds only for the continuous case. Thus it is not so hard to imagine that optimal routing or asymptotically convergent routing manages to lose its practical meaning and instead feasibility or availability of network resources such as link capacities and nodal buffers plays more important roles in making routing decisions. In reality, we have seen in many mathematical programming literatures that for the problems in this catagory, how to find a solution satisfying the required specification becomes rather more important an issue than how to improve the solutions to optimum.

3. The Routing Algorithm

The key point of our algorithm is the introduction of a new powerful penalty system which will be shown to handle effectively the intractable discreteness. This penalty system assigns much higher penalties to those free paths which, if chosen, will only increase the congestion levels of some already highly utilized links. At each iteration, the algorithm tries to find out a path which prefers the consumption(use) of then underutilized resources(links), so that the utilization levels of all network resources are eventually well-balanced. In order to achieve this desired state, it introduces two generalized concepts: goal utilization level and overutilization measure, the latter of which is a generalization of the infeasibility measure often used in mathematical programming (see [4] for precise definition). Our goal may then be stated to minimize the overall overutilization measure so as to keep the utilization level of each link below the prespecified goal utilization level.

Our solution procedure is composed of two phases, phase A and B; both are conceptually quite similar each other, but different types of penalities are employed. Phase A tentatively assigns paths, one for each origin-destination (simply, *o-d*) pair based on the current link utilization levels. If the path assignment of Phase A does not meet the requirement that all utilization levels should be below our goal utilization level, then Phase B is activated; say, sequential exchange operations are performed.

3.1. Phase A

At each iteration of Phase A, a path is assigned to one *o-d* node pair (or one set S_k), so that there are a total of K iterations in Phase A. Let us call an *o-d* node pair assigned(nonassigned) if a path has been assigned (nonassigned). Then by the end of iteration t $(t=1,2,..,K)$, we have t assigned pairs. Define the following notations:

ξ : the prespecified goal utilization level,
$K(t)$: the set of assigned *o-d* node pairs at the end of iteration $t\,(t=1,2,\dots,K)$,
$NK(t)$: the set of nonassigned node pairs by the end of iteration t; $NK(t) = K \setminus K(t)$,
$j(k)$: the path selected (or assigned) for *o-d* pair k, $k \in K$,
$S(t)$: the set of the paths assigned by the end of iteration t ,
$NS(t)$: the set of (free) paths not assigned by the end of iteration t; $NS(t) = S \setminus S(t)$,

With the above notations, we define a set function $a_i(K(t))$ as follows:

$$a_i(K(t)) = \sum_{k \in K(t)} a_{ij(k)} \qquad \text{for } i \in L \tag{10}$$

Note that $a_i(K(t))$ represents the $i-th$ rowsum of the selected paths, one for each of the assigned pairs in $K(t)$, at the end of iteration t. By the $i-th$ rowsum of the selected paths, we mean the sum of the $i-th$ entries of the corresponding columns in the matrix A. Thus our requirement is now reduced to

$$s_i(\xi) = \xi - a_i(K(K)) \geqq 0 \quad \text{for all } i \in L \tag{11}$$

Note here that $s_i(1) = s_i$ for all i and the feasibility condition of P corresponds to (11) with $\xi = 1$.

Iteration t is to find the path with minimum penalty among the paths belonging to $(K\text{-}t)$ non-assigned pairs, *i. e.*, $NS(t)$, and to assign that path to the corresponding *o-d* pair. The penalty $P_j(t)$ defined for path j in $NS(t)$ at iteration t is given as follows:

$$P_j(1) = \sum_{i \in L} a_{ij}, \quad j \in S \tag{12a}$$

$$P_j(t) = \sum_{i \in L} a_i(K(t-1))a_{ij}, \qquad j \in NS(t),\ t = 2, \ldots, K \tag{12b}$$

Note that $P_j(1)$ is the sum of entries in column j of the matrix A, which will be simply called the colsum of path j. This may be interpreted as the traffic load imposed on the network by setting up path j. Likewise, $P_j(t)$ is a weighted colsum of path j where the $i-th$ weight is the $i-th$ rowsum of the already selected paths. Also this may be interpreted as weighted traffic load imposed on the network by setting up path j.

The procedure for Phase A

Initialization: Let $K(0) = S(0) = \emptyset$. Also let $t = 1$ and compute $P_j(1)$ of (12) for $j \in NS(0)\ (= S)$

Interation t $(t = 1, 2, \ldots, K)$:

1. Let $J(t) = \{\ p \in NS(t-1) \mid P_p(t) = \underset{j \in NS(t-1)}{\text{minimum}}\ P_j(t)\ \}$ (13)

   If $|J(t)| = 1$, let $j^* = p$. Otherwise, get

   $JJ(t) = \{\ q \in J(t) \mid P_q(1) = \underset{j \in J(t)}{\text{minimum}}\ P_j(1)\ \}$ (14)

   If $|JJ(t)| = 1$, let $j^* = q$. Otherwise, let j^* be any q in $JJ(t)$.
2. Let $k^* \in NK(t-1)$ be such that $j^* \in S_{k^*}$. Then $K(t) = K(t-1) \cup \{k^*\}$ and $j(k^*) = j^*$. Also, $a_i(K(t)) = a_i(K(t-1)) + a_{ij^*}$ for all $i \in L$. If $t = K$, go to 3. Otherwise, compute $P_j(t)$ for all $j \in NS(t)$ and go to the next iteration with $t = t + 1$.
3. If $s_i(\xi)$ defined in (11) is nonnegative for all $i \in L$, a satisfactory solution is obtained. Otherwise, go to Phase B.

In light of the interpretation of $P_j(t)$, the above procedure chooses paths sequentially in the increasing order of their marginal contributions to the cumulative traffic load, hoping for achieving the minimum overall traffic load. Specifically, (14) along with (13) reflects this philosophy in that the path with the smallest net load is most preferred if tie occurs when choosing the smallest weighted traffic load.

3.2. Phase B

Assuming we have an incumbent selection $y=(j(1),\ldots,j(K))$, define the overutilization measure $O(y,\ \xi)$ as a function of y and the goal utilization level ξ. That is,

$$O(y,\ \xi) = \sum_{i\in L} \max\ \{0, -s_i(\xi)\} \tag{15}$$

With this definition, Phase B may be viewed as a sequence of operations exchanging, one by one, some incumbent path for the one which will significantly decrease the overutilization measure while keeping the increase of other rowsums at minimum level.

To proceed further, we decompose the link set L into LO, LU and LS according to their utilization level as follows:

$$LO = \{ i\in L \mid s_i(\xi) < 0 \} \tag{16a}$$
$$LU = \{ i\in L \mid s_i(\xi) > 0 \} \tag{16b}$$
$$LS = \{ i\in L \mid s_i(\xi) = 0 \} \tag{16c}$$

Then, with respect to our goal utilization level, the links in LO, LU and LS may be said to be 'overutilized', 'underutilized' and 'saturated' respectively.

Now define the following: for each $j\in S_k$ and $k\in K$,

$$\alpha_{ij} = a_{ij(k)} - a_{ij},\quad i\in L \tag{17}$$
$$R_j = \sum_{i\in L} \max\ \{0, -s_i(\xi)\}\ \max\ \{0, \alpha_{ij}\} \tag{18}$$
$$= \sum_{i\in LO} -s_i(\xi)\ \max\ \{0, \alpha_{ij}\}$$
$$C_j = \sum_{i\in L} a_i(K(K))\max\ \{0, -\alpha_{ij}\} \tag{19}$$

R_j represents the decrease in overutilization measure of the solution, weighted on $\max\ \{0,-s_i(\xi)\}$ for $i\in L$, when path j, $j\in S_k$, is substituted for path $j(k)$. Note that each term of (18) takes positive value only when both the conditions $i\in LO$ and $\alpha_{ij}>0$ are met. C_j, though looking different, is in fact the same with $P_j(t)$'s in Phase A in its underlying notion. It represents the increase in rowsums, weighted on $a_i(K(K))$ for each $i\in L$, when path j, $j\in S_k$, is substituted for path $j(k)$. In short, $R_j(C_j)$ may be viewed as the level of positive (negative) potential for path j.

Recall that Phase B, starting with an unsatisfactory selection, searches for a satisfactory one via a sequence of exchange operations. Though each measure can be used effectively by itself, their effects are multiplied by introducing the following integrated measure of ratio form. Namely,

$$NP_j = R_j/C_j \quad \text{for } j\in S \tag{20}$$

If $C_j=0$, then $NP_j=\infty$. This solidly reflects the net potential of path j. Before listing the procedure, define Q (a subset of S) as the set of paths, which, if substituted for the incumbent, will increase the rowsums of at least one already overutilized or saturated link. Say, if a path j in S_k happens to belong to Q, $\alpha_{ij}=a_{ij(k)}-a_{ij}<0$ for some i in LO or LS.

The procedure for Phase B

Step 0: Calculate $O(y,\ \xi)$ for the current incumbent selection y and let $O(\xi) = O(y,\ \xi)$.

Step 1 : Find out Q and obtain α_{ij} for all $i \in L$ and $j \in S \setminus Q$. Also, compute R_j, C_j and thus NP_j for all $j \in S \setminus Q$.

Step 2 : Find j^* such that

$$NP_{j^*} = \underset{j \in S \setminus Q}{maximum}\ NP_j \tag{21}$$

If tie occurs, choose any. Let $j^* \in S_{k^*}$. Update $j(k^*) = j^*$ and compute $s_i(\xi)$ for all $i \in L$. If $s_i(\xi) \geqq 0$ for all $i \in L$, then stop. Otherwise, go to Step 3.

Step 3 : Calculate the corresponding overutilization measure $O(y, \xi)$. If $O(y, \xi) < O(\xi)$, update $O(\xi)$ by $O(y, \xi)$ and reset the loop count $NLOOP = 1$ and go to Step 1. Otherwise increase $NLOOP$ by one and go to Step 4.

Step 4 : If $NLOOP > NMAX$, stop. Otherwise, go to Step 1 ($NMAX$ is a prespecified integer).

We note that Step 3 and 4 of Phase B are introduced to prevent Phase B from cycling. To show how they handle this undesirable phenomenon, consider the case where the goal utilization level is set too tight to yield a satisfactory solution. In this case, even when the best solution is already on hand, Step 1 and 2 of Phase B may occasionally be forced to move to a possibly worse solution. Without some protective device, the search in this fashion may result in cycling in the worst case. However, when the solution found is a mere local optimal solution, it is necessary to move temporarily to a worse solution in the hope of finding a far better solution at later iterations. An instance will be exhibited later in Section 5. Thus by controlling the value of NMAX, the limit of iterations to proceed without any decrease in the overutilization measure, we can trade an improved solution for some possible degradation in computational efficiency.

3.3. Algorithmic Variations

We have dealt with the prototype of the algorithm so far. This section considers some refinements of Phase A and B for incorporating a number of performance criteria mentioned in Section 2. These will include the use of specialized penalty system, the parametrization of the goal utilization level and finally the selection of performance measure used in the stopping rule of Step 3. Since most of Phase A and B remain effective, we will outline those modified parts only.

Firstly, the expression (12b) and (19) can be further refined to reflect more precisely the properties of the involved objective function. For instance, when we are interested in minimizing the average message delay expressed by the equation (1), its first order information can be effectively utilized to define the link weights used to determine the penalties for candidate paths; that is, the first order derivative of $g(\rho)$ evaluated at the current link utilization levels are more effective in reflecting the convexity of $g(\rho)$.

Second variation is the parametrization of ξ. It may be statically fixed or dynamically adjusted. If we prespecify ξ at some value usually smaller than one, our system goal is to enforce the link utilization levels below the fixed value. In particular, feasibility itself becomes our major concern if ξ is set to one. By dynamic adjustment, we mean that ξ should be adjusted to a value depending on the current state of link utilization. For example, if we set ξ to the second largest link utilization level at each iteration, the cardinality of the set LO is always one and the corresponding system goal is to minimize the maximum link utilization level.

The last one is the selection of an appropriate stopping measure in Step 3 of Phase *B*. If our goal is to minimize the average message delay, $O(y, \xi)$ as a stopping measure should be replaced accordingly by the average message delay. The same is true for the minimization of maximum link utilization level or the maximum link delay.

4. Network Realization

We started with the assumption that there exist a prespecified set of communication paths for each *o-d* node pair, which holds for an SNA-like networking environment. A close inspection of the algorithm shows that this assumption can be dropped for its application. Then, the algorithm turns out to be none other than a sequential identification of shortest paths. To be specific, Phase *A* may be described as follows:

Iteration $t (t = 1, 2, \ldots, K)$:

Find the shortest path for each *o-d* pair in $NK(t-1)$, when the distance defined for link i is $a_i(K(t-1))a_{ij}$ (see the definitions (3) and (10) of a_{ij} and $a_i(K(t-1))$ respectively). Identify the path with the shortest distance among $(K-t+1)$ such paths and assign that to the corresponding pair in $K(t-1)$.

The similar type of description can be applied to Phase B. Assume we have a tentative assignment of paths, whether from Phase A or some other routing scheme. Then we have at hand three subsets of the link set, LO, LU and LS as defined by (16). Remind that the strategy of Phase *B* is to reroute some sessions connected through the congested (or more precisely, 'overutilized' in our terms) links, so that underutilized resources (links) take over those sessions.

Let $KO = \{ k \in K \mid \text{the path } j(k) \text{ traverses at least one link in } LO \}$. (22)

Then disconnecting or rerouting the sessions for any *o-d* pair of KO is likely to alleviate the traffic load of the links in LO to some extent. Note that all the sessions for each *o-d* pair are required to pass through a single path. The sessions or equivalently the selected paths, one for each *o-d* pair of KO, are the candidates for rerouting.

Define also

$$LO(k) = \{ i \in LO \mid \text{the link } i \text{ is on the path } j(k) \} \tag{23}$$

The amount of possible degradation in overutilization measure can then be estimated as the following weighted sum.

$$\overline{R}_k = \triangle_k \sum_{i \in LO(k)} (-s_i(\xi)/C_i) \qquad \text{for } k \in KO \tag{24}$$

Note that this differs from R_j of (18) in that excluded was the dependence of expression (18) on the path j in S_k. Furthermore, it can be easily seen from (18) and (24) that the following relation holds between them.

$$R_j \leq \overline{R}_k \quad \text{for all } j \in S_k \tag{25}$$

Now we turn our attention to C_j of (19). Note here that maximizing the net potential NP_j over $j \in S_k$ is equivalent to minimizing C_j over the same domain when R_j's over all j in S_k are replaced by a common value $\overline{R}_k$. With this notion in mind, we consider the subgraph $G(N, LU)$

which is composed of underutilized links only. Let l_k be the shortest distance for the k-th o-d node pair on the subgraph $G(N, LU)$ with the link distances $d_i(k)$ defined by:

$$d_i(k) = \begin{cases} 0 & \text{if the link } i \text{ is on the path } j(k) \\ (\xi - s_i(\xi))/C_i & \text{otherwise} \end{cases} \tag{26}$$

Then the net potential NP_j attains its maximum over the set S_k at this shortest path with distance l_k and its correspondent of C_j becomes simply $l_k \Delta_k$. Our final measure NP_k, which also depends only on the index k for $k \in K$, is now at hand.

$$NP_k = (1/l_k) \sum_{i \in LO(k)} (-s_i(\xi)/C_i) \quad \text{for } k \in KO \tag{27}$$

Thus the network realization of Phase B follows:

Network realization of Phase B

The relevant steps are:

Step 1': Identify LO, LU, LS, KO and $LO(k)$ for each k in KO and compute $\bar{R}_k$ as defined by (24). Consider the subgraph $G(N, LU)$ to find the shortest path l_k for each $k \in KO$ and finally compute NP_k following the definition (27).

Step 2': Find k^* in KO such that

$$NP_{k^*} = \underset{k \in KO}{\text{maximum}}\ NP_k \tag{28}$$

Substitute the path $j(k^*)$ for the one of distance l_{k^*}.

If the updated values $s_i(\xi)$ are nonnegative for all $i \in L$, then stop. Otherwise, go to Step 3.

5. Numerical Examples

In order to see how Phase A and B operate, let us consider a network shown in Figure 1, where $N = \{1, 2, 3, 4\}$, $L = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $K = \{1, 2, 3, 4\}$ and the figures in the parentheses denote the corresponding link capacities in kbps.

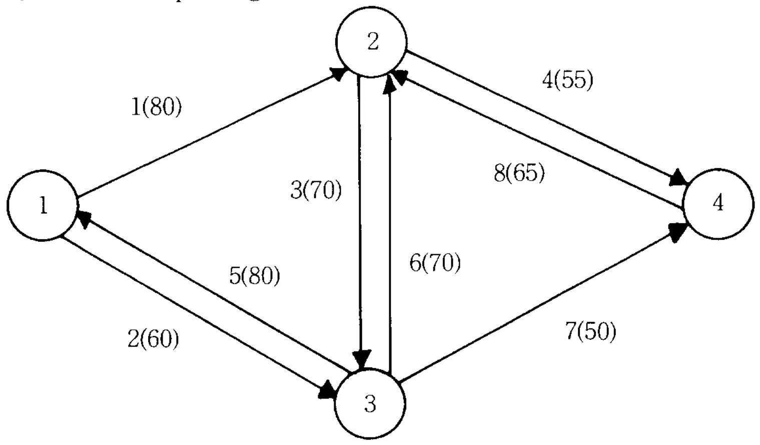

Figure 1. Network configuration 1

This example is introduced mainly for exposition purpose so that it may be far from the real-world situation, but suffices to show how the operations of Phase *A* and *B* are performed. The related input data are summarized in Table 1a and the corresponding *A* matrix is shown in Table 1b.

With these data, Phase *A* selects paths one by one for each *o-d* pair. Table 2 shows some related data and the resultant path selection {1, 4, 9, 11}.

Table 1a. Summary of input data

k	o-d pair	Δ_k^*	no. of paths	path no.	path as a sequence of links
1	1→2	30	3	1	1
				2	2 6
				3	2 7 8
2	1→4	25	4	4	1 4
				5	2 7
				6	1 3 7
				7	2 4 6
3	3→4	20	2	8	7
				9	4 6
4	3→2	35	3	10	6
				11	1 5
				12	7 8

* The unit is messages/sec and the average message length $1/\mu = 1.3$ (kbit/message) is used.

Table 1b. The *A* matrix

link \ path	1	2	3	4	5	6	7	8	9	10	11	12
1	.488			.406		.406					.569	
2		.650	.650		.542		.542					
3						.464						
4				.591			.591		.473			
5											.569	
6		.557					.464		.371	.650		
7			.780		.650	.650		.520				.910
8			.600									.700

Table 2. Iterations of Phase A

Iteration	Path no.	Current utilization levels							
1	1	.49	.0	.0	.0	.0	.0	.0	.0
2	4	.89	.0	.0	.59	.0	.0	.0	.0
3	9	.89	.0	.0	1.06	.0	.37	.0	.0
4	11	1.46	.0	.0	1.06	.57	.37	.0	.0

In Table 2, the path selection {1, 4, 9, 11} of Phase A have two overutilized links 1 and 4. If our goal is to find a feasible selection, we should go further to Phase B to find such a selection. Table 3 summarizes the iterations of Phase B.

Table 3a. Iteration 1 of Phase B

Path j	R_j	C_j	NP_j	Path selection
2	.225	.207	1.09	$\max\{NP_j\}=NP_3=\infty$
3	.225	.0	∞	

At iteration 1, NP_3 is the maximum so that path 3 replaces path 1 and the resultant utilization levels become (.97 .65 .0 1.06 .57 .37 .78 .60). Since the utilization factor of link 4 is 1.06, we go further to the next iteration.

Table 3b. Iteration 2 of Phase B

Path j	R_j	C_j	NP_j	Path selection
1	.0	.475	.0	$\max\{NP_j\}=NP_6$
2	.0	.207	.0	$=NP_8$
5	.038	.859	.044	$=.074$
6	.038	.507	.074	
7	.0	.525	.0	
8	.030	.406	.074	
10	.0	.241	.0	
12	.0	1.130	.0	

At iteration 2, $NP_6=NP_8=.074$. Since $P_6(1)=1.52$ and $P_8(1)=0.52$, path 8 is preferable to path 6. As a result, the updated path selection is {3, 4, 8, 11} and the resultant utilization levels become (.97 .65 .0 .59 .57 .0 1.30 .60). Since our goal does not have been attained yet, iteration 3 follows.

Table 3c. Iteration 3 of Phase B

Path j	R_j	C_j	NP_j	Path selection
1	.234	.475	.492	
2	.234	.0	∞	$\max\{NP_j\} = NP_2 = \infty$

As a result of iteration 3, a better path selection {2, 4, 8, 11} is obtained with the utilization levels (.97 .65 .0 .59 .57 .56 .52 .0). Since all of these utilization levels are below 1, the selec- {2, 4, 8, 11} is feasible and the procedure stops.

We add some remarks on the example. By enumerating all combinations of path selection, we have found that among the total 72 combinations, only two are feasible, which are shown in Table 4.

Table 4. Two feasible selections

Path selection	Utilization levels							
{2, 4, 8, 11}	.97	.65	.0	.59	.57	.56	.52	.0
{2, 6, 9, 11}	.97	.65	.46	.47	.57	.93	.65	.0

Apparently, the former selection is better than the latter since the latter is dominated by it in all links except link 4. Thus it can be said that our algorithm has succeeded in finding a feasible selection {2, 4, 8, 11}, which is the better one of the only two feasible selections the problem has among the total 72 possible combinations.

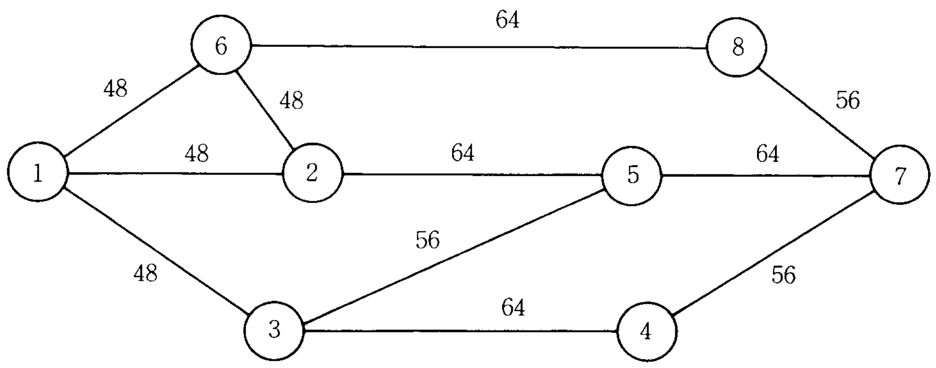

Figure 2. Network configuration 2

In order to estimate the computational efficiency of our algorithm, a more practical sized network as shown in Figure 2 was constructed and the algorithm was tested with this problem. The problem statistics are as follows: the number of nodes is 8, the number of directed links is 22, the number of *o-d* node pairs is 28 and the total number of candidate paths enumerated is 128. The results obtained from the computer run performed on KAIST's cyber 174-16 with the code written in FORTRAN IV are summerized in Table 5.

Table 5. Summary results

Problem	Average mess. length	*NMAX*	ξ	$O(y, \xi)$	*CPU* times in seconds**
1	. 9*	3	1. 00	. 0	. 154
		3	. 90	. 0	. 156
		3	. 80	. 102	. 206
2	. 7*	3	. 80	. 0	. 156
		3	. 70	. 0	. 154
		3	. 60	. 125	. 210

* The unit is 1000 bits/message.
**This does not include *I/O* times.

5. Conclusion

This paper has suggested a routing or route selection algorithm which is applicable for both design and implementation purposes. Its effectiveness in finding satisfactory solutions in the intractable discrete solution space seems to be attributable to its new powerful penalty system. It, though developed mainly for design and analysis purposes, can also form the basis for developing routing protocols in session-based operating environment, particularly when the traffics are of bulk-type so that continuity assumption of solution space no longer holds valid. Furthermore, the philosophy of Phase *B* makes room for its use in partial flow control ; that is, if Phase *B* can not find a path with sufficient bandwidth for a particular session or group of sessions, the corresponding flow rate is controlled by dropping some of them or by reducing the transmission rate of message for each session.

As a final remark, we would like to point out that its usage may be enlarged to determine the primary routes of IBM's SNA networks and also to provide a good initial flow configuration in any flow assignment problem which is a subproblem of global topological optimization, and finally even in the areas of investment, scheduling and allocation, where the type of decisions is choosing one from a finite number of alternatives available to the decision maker. It is expected that for many such cases, the concept of penalties can be equally powerfully utilized as in our problem.

References

1. Chang, S. G. and D. W. Tcha, "A Heuristic for Multiple Choice Programming," *Comput. & Ops. Res.*, Vol. 12, No. 1, pp. 25-37, 1985.
2. Chang, S. G., Routing and Flow Assignment in a Distributed Communication Network : Algorithms and Sensitivity Analysis, Ph. D. Dissertation, Dept. of Management Science, KAIST, 1984.
3. Fratta, L., M. Gerla and L. Kleinrock, "The Flow Deviation Method-An Approach to Store-and-Forward Communication Network Design," *Networks,* Vol. 3, pp. 97-133, 1973.
4. Garfinkel,R. and G. Nemhauser, *Integer Programming,* John Wiley & Sons, 1972.
5. Gavish, B. and S. Hantler, "An Algorithm for Optimal Route Selection in SNA Networks," *IEEE Trans. on Comm.,* Vol. COM-31, No. 10, pp. 1154-1161, 1983.
6. Golestaani, S. J., "A Unified Theory of Flow Control and Routing in Data Communication Networks," Report LIDS-TH-963, MIT, January 1980.
7. Kleinrock, L., *Queueing Systems Volumn II: Computer Applications,* John Wiley & Sons, 1976.
8. Maruyama, K. and D. Shorter, "Dynamic Route Selection Algorithms for Session-based Communication Networks," *ACM proceedings,* pp. 162-169, 1983.
9. Tcha. D. W. and K. Maruyama, "On the Selection of Primary Paths for a Communication Network," *Computer Networks and ISDN Systems,* Vol. 9, No. 4, pp. 257-265, 1985.
10. Toyoda, Y., "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-One Programming Problems," *Management Science,* Vol. 21, pp. 1417-1427, 1975.
11. Weintraub, A. and J. Gonzalez, "An Algorithm for the Traffic Assignment Problem," *Networks,* Vol. 10, pp. 197-209, 1980.