The Optimal Virtual Path Design of ATM Networks

Zhigang Qin*, Tetiana Lo**, Felix F. Wu**

*Optimal Networks, Palo Alto, CA 94303

**Department of Electrical Engineering and Computer Sciences

University of California, Berkeley, CA 94720

Abstract

This paper studies the optimal configuration of virtual paths in ATM networks. A linear model that minimizes the maximum flow density of links is proposed for the optimal virtual path configuration. The advantage of the modeling is that traffic loads are distributed evenly over entire network. An algorithm based on the multi-commodity approach is proposed to solve the problem. Examples are given to show that this algorithm solves the optimal VP assignment problem very quickly and efficiently. Since only single source-destination linear programming subproblems need to be solved in our approach, this algorithm can be applied to large-scale networks.

Keywords: *ATM, Virtual Path*

1 Introduction

The virtual path (VP) concept is a useful and powerful transport and management mechanism for Asynchronous Transfer Mode (ATM) networks. A virtual path consists of a bundle of virtual channels (VCs, also referred to as virtual circuits) which can be treated as a single large unit instead of individual units. Virtual channel connections are set up for end-to-end connections of end users, usually between two AAL (ATM Adaptation Layer) entities. Virtual path connections are set up to facilitate the flow of traffic through the network. Virtual channels and virtual paths are identified by virtual path identifiers (VPI) and virtual channel identifiers (VCI) respectively. Each bundle of VCs possesses the same VPI and the same end-points. Virtual path connections can be set up permanently, for example, signaling VPs, or set up dynamically and can be updated whenever the traffic patterns have changed. Virtual channel connections (VCC) usually are set up in real time. When a new call arrives, a virtual channel connection must be established before the traffic can be delivered to the other end. The VC connection will be torn down after the VC session is over.

Switching can be performed on one of two levels: the virtual path level or virtual channel level. The corresponding switching devices at each level are cross-connects and ATM switches. For virtual path connections, the switching functions are based upon the VPI values only; while for virtual channel connections, the switching functions are based on both VPI and VCI values. Also, statistical multiplexing is implemented in switching functions on the virtual channel level. Therefore, ATM switches are much more complex and expensive than cross-connects. At each cross-connect, each incoming VPI is mapped to an outgoing VPI; at each ATM switch, both the VPI and VCI of a connection are mapped to the outgoing VPI and VCI according to a routing table as shown in Fig. 1. The VPI and VCI are determined locally.

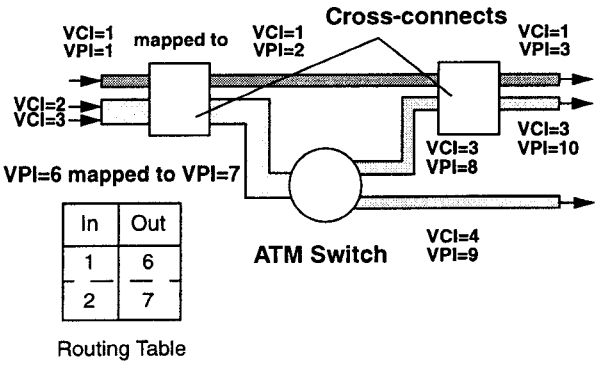


Fig. 1. VPI/VCI Routing in ATM Networks

The advantage of virtual paths is the simplification of call setup (signaling), routing, and call admission control. Another advantage is the logical network abstraction created by virtual path routing and the cross-connect technique in ATM networks. The cross-connect function allows traffic on a virtual path to be routed without being statistically multiplexed with traffic on other VPs, thereby ensuring quality of service (QOS) at the cell layer. Thus, in virtual path routing, there is no need to check network

resources to verify the QOS requirement at each hop along a virtual path, *i.e.,* a virtual path is equivalent to a one-hop path for delivering ATM traffic. For this reason, virtual paths can be used to form a logical network, and to reconfigure the network topology, as shown in Fig. 2, where a

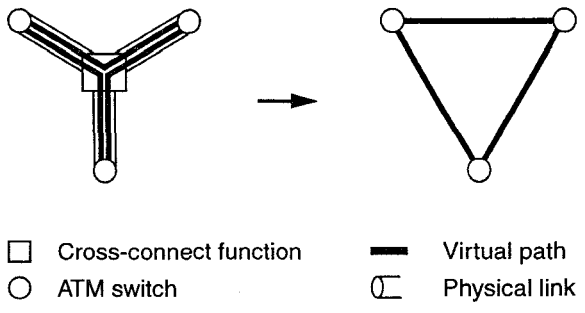


Fig. 2. Illustration of Logical Network Design

four-node network is re-configured as a three-node completely connected network. Reconfiguration of the network allows one to change the topology of the network dynamically, so as to best accommodate traffic patterns. As indicated in [4], the use of virtual paths can save more than 90% processing time when compared with virtual circuit routing. In the following sections, we study the optimal reconfiguration of logical networks.

2 Optimal Reconfiguration Modeling

In this section, we model the logical network design problem as a multicommodity flow problem and propose to solve the problem by using decomposition/coordination techniques. The problem can be stated as follows:

Given:

- the physical network (topology and link capacities),
- traffic requirements,

Objective:

maximize network performance.

Subject to:

- physical link capacity constraints,
- flow constraints,
- the traffic requirements,
- fairness constraints,
- the call-setup delay requirement.

Determine:

- virtual path assignment and bandwidth allocation.

Virtual paths have been studied by many researchers to improve ATM network performance [2]-[9]. In [2], the authors propose a virtual path layout optimization model, heuristic search algorithms to find the VP layout, and a bandwidth assignment method. Guidelines for the design of robust VP layouts were also presented. In [3], a strongly connected, directed graph was used to represent the network topology, and it was proved that optimizing the use of virtual paths involves computationally-hard problems for the model they used. Thus, approximation solutions to establish VPs were developed. In [6] and [7], virtual path assignment and virtual circuit routing problem were jointly investigated. Due to computational complexity, only heuristic algorithms were proposed to solve this joint optimization problem. In [9], it was assumed that the virtual path layout was given, and only bandwidth allocation methods were studied. In summary, due to the complex nature of the VP optimization problem, only heuristic algorithms have been proposed.

In this study, we propose a less complex model for VP layout optimization and bandwidth allocation with fairness constraints, and solve the problem using a multicommodity network flow algorithm. To consider the fairness constraints, we define the objective as maximizing the throughput of the network without penalizing traffic with long routes, *i.e.*, to minimize the maximum traffic density of links. In order to specify the traffic requirements, the effective bandwidth concept is used [1]. Effective bandwidth provides a unified metric representing the effective load for the connection. It is calculated based on the source statistics and requested QOS of the connection. Since one virtual path is equivalent to a one-hop path, the effective bandwidth does not change along the virtual path. In our scheme, we attempt to assign each source-destination pair a virtual path. Thus, effective bandwidth provides a satisfactory description of the traffic in our model.

To formulate the logical network design problem, we define the following variables:

Given:

N: the set of all nodes,

S: the set of all source nodes,

D: the set of all destination nodes,

d : processing delay in a cross-connect node,

Δ : a very large penalty number,

M_{sd} : the maximum call-setup delay for a sd pair,

N_v : the largest VPI number,

L_{ij} : capacity of physical link ij,

t^{sd} : the traffic requirements for source-destination pair sd,

X_p^{sd} : the set of ATM switches along a VP p of source-destination pair sd.

To solve:

f_{ij}^{sd} : the amount of flows for source destination pair sd on link ij,

$$z_{ij}^{sd} = \begin{cases} 0 \text{ if virtual path for } sd \text{ does not include link } ij \\ 1 \text{ if virtual path for } sd \text{ includes link } ij \end{cases}$$

ϕ^{sd} : the slack variable.

Based on the above definitions, we formulate the problem as follows:

$$min\left(\max_{i>j}\left(\sum_{sd}\langle z_{ij}^{sd} f_{ij}^{sd} + z_{ji}^{sd} f_{ji}^{sd}\rangle / L_{ij}\right) + \Delta \cdot \phi^{sd}\right) \quad (1)$$

subject to

$$\sum_{sd}\langle f_{ij}^{sd} + f_{ji}^{sd}\rangle \le L_{ij} \qquad i, j \in N \quad (2)$$

$$\sum_j f_{ij}^{sd} - \sum_j f_{ji}^{sd} = \begin{cases} t^{sd} - \phi^{sd} & i \in S \\ -t^{sd} + \phi^{sd} & i \in D \\ 0 & \text{otherwise} \end{cases} \text{ for all } \begin{matrix} s \in S \\ d \in D \end{matrix} \quad (3)$$

$$\sum_{sd}\sum_j z_{ij}^{sd} \le N_v \qquad i \in N \quad (4)$$

$$\sum_{sd}\sum_i z_{ij}^{sd} \le N_v \qquad j \in N \quad (5)$$

$$\sum_{i \in X_p^{sd}} \sum_{j \in X_p^{sd}} z_{ij}^{sd} d \le (1/2) M_{sd} \qquad s \in S, d \in D \quad (6)$$

The objective (1) is to minimize the maximum traffic density; thus to distribute the loads evenly throughout the entire network. The objective function treats the source-destination pairs with long routes and those with short routes equally, and attempts to make full use of the network resources. Since traffic requirements may exceed the capacity of the network, slack variables are introduced in the traffic requirement constraints (3). In the objective function, we assign a large penalty coefficient to the slack variables. Thus the slack variables are forced to take on zero values if there is sufficient network capacity. Equation (2) is the capacity constraint. It requires that the flow on a specific link be less than the physical link bandwidth. Equation (3) consists of the flow conservation equations. It also requires that all traffic requests be accommodated by virtual paths if there is sufficient bandwidth. Equations (4) and (5) state that the total number of VPs at a node should not exceed the VPI limit at the node. The total number of VPs at a node is the summation of all VPs leaving or entering that node. Since the number of VPs entering a node may differ from those leaving the node, both numbers must be checked to ensure that the constraint on the maximum number of VPs at each node is not violated. Equation (6) specifies that the call-setup time cannot exceed a specified limit. Since the main delay in the call-setup is the processing time in ATM switches, we approximate the call-setup time by the number of VP hop counts in a VCC connection.

In summary, the VP design problem is formulated as a nonlinear programming problem. In the next section, we present an algorithm using the multicommodity technique to solve this problem.

3 Algorithm

Since constraints (5) and (6) are very loose constraints, we ignore these two constraints in the first stage of solving the problem, and verify their validity after finding an optimal solution. If the constraints have been violated, measures will be taken to satisfy the constraints. The details of the algorithm are presented as follows.

Since we minimize the flow densities of the links, the flow variables will take on the smallest feasible values. Thus, the physical capacity constraints are actually considered implicitly in the objective function. Therefore, we can remove the physical constraints from the formulation, creating the following optimization problem:

$$min\left(\max_{i>j}\left(\sum_{sd}(f_{ij}^{sd} + f_{ji}^{sd})/L_{ij}\right)\right) \quad (7)$$

subject to

$$\sum_j f_{ij}^{sd} - \sum_j f_{ji}^{sd} = \begin{cases} t^{sd} & i \in S \\ -t^{sd} & i \in D \\ 0 & \text{otherwise} \end{cases} \text{ for all } \begin{matrix} s \in S \\ d \in D \end{matrix} \quad (8)$$

Note that constraint (2) has been removed and is taken care of implicitly in (7) by minimizing the flow densities of links. Since the constraints for this new formulation are decoupled, we can decompose the problem into n (number of source-destination pairs) single source-destination pair linear programming subproblems that can be solved by any linear programming solver.

Let z be a new variable that satisfies the following inequalities,

$$z \ge \sum_{sd}(f_{ij}^{sd} + f_{ji}^{sd})/L_{ij} \qquad i > j;\ i, j \in N \quad (9)$$

Then the entire problem can be rewritten as:

$$min\left((z) + \sum_{i>j} \mu_{ij}\left(\sum_{sd}(f_{ij}^{sd} + f_{ji}^{sd})/L_{ij} - z\right)\right) \quad (10)$$

subject to

$$\sum_j f_{ij}^{sd} - \sum_j f_{ji}^{sd} = \begin{cases} t^{sd} & i \in S \\ -t^{sd} & i \in D \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } \begin{matrix} s \in S \\ d \in D \end{matrix} \quad (11)$$

where μ_{ij} is the Lagrangian multiplier. Based on this formulation, we decompose this problem into the following subproblems:

Let l represent the source-destination pair, $(q+1)$ be the current number of the iteration, then the subproblem is formulated as:

$$min\left(\max_{i>j}(\lambda_{ij}^q((f_{ij}^l + f_{ji}^l)/L_{ij})\right) \quad (12)$$

subject to

$$\sum_j f_{ij}^l - \sum_j f_{ji}^l = \begin{cases} t^l & i \in S \\ -t^l & i \in D \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

where λ_{ij}^q is the penalty factor at the qth iteration. The subproblem formulation is only a single source-destination problem which can be solved easily. From the solutions of the subproblems we can obtain the flow on a link by adding the flows of subproblems which go through this link. Once we obtain the flows on all links, we can improve the solution by updating the penalty factors using following equation:

$$\lambda_{ij}^{q+1} = [\lambda_{ij}^q + \theta^q((f_{ij}^l + f_{ji}^l)/L_{ij} - A^q)] \quad (14)$$

where A^q is the average flow density, θ^q is a step size specifying how far we move from the current solution λ^q. We use 1 in our algorithm. If the density of a link is higher that the average density, we add a penalty factor proportional to the difference between these two density values. Thus, at the next iteration, all subproblems will attend to reduce the amount of flow on these links if alternative routes have lower flow densities.

The initial values of λ^q are set to be 1, and the initial flows of all links, $f_{ij}^{0,sd} + f_{ji}^{0,sd}$, are determined by minimizing the total flow of the following problem for each source-destination pair *sd*,

$$\min \sum_{i>j} \langle f_{ij}^{sd} + f_{ji}^{sd} \rangle \quad (15)$$

subject to

$$\sum_{i>j} \langle f_{ij}^{sd} + f_{ji}^{sd} \rangle \le L_{ij} \quad (16)$$

$$\sum_j f_{ij}^{sd} - \sum_j f_{ji}^{sd} = \begin{cases} t^{sd} & i \in S \\ -t^{sd} & i \in D \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

With this new formulation, we can apply a linear programming solver to solve the subproblems. We then update the penalty factors with the solutions of the subproblems, and move to get the next subproblem. This process is repeated until the solutions to all subproblems have converged. The block diagram representation of the algorithm is shown in Fig. 3.

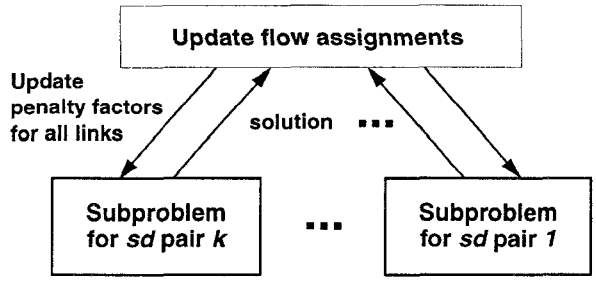


Fig. 3. Structure of the Virtual Path Assignment Algorithm

In the above formulation we do not assume that sufficient network capacity is available to satisfy all traffic requirements. For the case that the traffic requirements exceed the network capacity, the above algorithm can still be used, because the link capacity constraints are considered in the objective function. However, some links will be assigned flows with a total bandwidth exceeding their capacities. Since VP bandwidths are assigned based on the flows obtained from the solution, it is necessary to reduce the bandwidths of the VPs passing through the links proportionally such that the capacity constraints of these links will be satisfied.

Although the VPI constraint is not a tight constraint, for a large ATM network, the number of virtual paths needed may exceed the VPI limit. If this is the case, a heuristic algorithm that merges the virtual paths through the switches where the VPI number limit is violated is used to

solve this problem. Details of the merge operation are discussed in the next paragraph. The connection setup time constraint also needs to be validated. The main task in setting up a virtual channel connection is resource-checking at the ATM switches. Therefore, the connection setup delay mainly depends on the number of ATM switches along the connection. Other delays, such as propagation delay, transmission delay, *etc.*, are relatively insignificant compared to the delay of resource checking at the switches. For this reason, we approximate the connection setup delay constraint as a constraint on the number of ATM switches or VP hop counts on a VCC connection. The number of VP hop counts will be increased by one if a merge operation is performed on the VPs. Thus when the merge operation is performed, the connection-setup time constraint is verified simultaneously. If the merger operation causes a violation of the call-setup time constraint on a particular VP, other VPs that pass through the switch for VP merging must be chosen. This process is repeated until either feasible virtual path assignments are found or no feasible solution can be obtained. In the latter case, the call setup time constraint must be relaxed.

Using the merge operation to reduce the number of VPs when the VPI limit is violated is first proposed in [2]. While the procedures we use for merging VP are the same as those in [2], a different set of criteria for choosing VPs for merging is presented. The criteria we propose are: 1) call setup constraints; the number of hop counts along the resulting VPs through the merge operation should not exceed the maximum number of VP hop counts allowed. 2) multiplexing gain; since the call-blocking function of bandwidth is convex, we can obtain higher multiplexing gain (lower call blocking rate) by adding the same amount of bandwidth on the smaller-bandwidth VPs versus the larger-bandwidth VPs. Thus, smaller-bandwidth VPs are preferred for the merging operation. The merge operation procedure is as follows: assume that the VPI constraint is violated at switch S for the incoming VPs. 1) Select the two VPs with the least amount of bandwidth that are both incoming to node S and have not been selected before. 2) For all source-destination pairs to which the two VPs are assigned, determine whether the merge operation will result in a violation of the call-setup time constraints. If so, go back to step 1, otherwise, continue. 3) Locate the portions of these two VPs that share the same nodes and include the node S. 4) Merge the two parts of these two VPs as illustrated in Fig. 4. This process is repeated until no node has a VPI number which exceeds the limit.

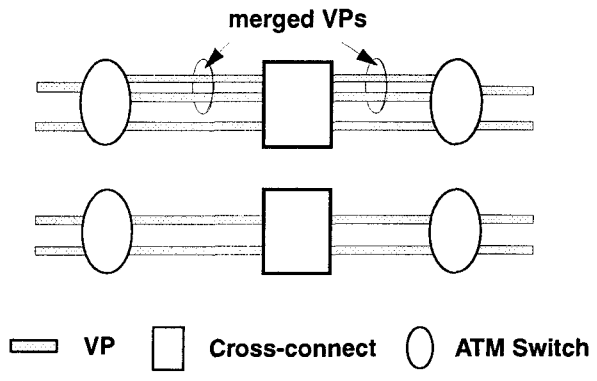


Fig. 4. Illustration of VP Merge Operations

After one merge operation, the VPI numbers of all nodes along the merged portion are reduced. The amount by which a VPI number is reduced differs for end nodes and intermediate nodes. Assume that two VPs pass through the same set of nodes, $n_0, n_1, \ldots n_m$. The merging of these two VPs will reduce the outgoing number of VPs at node n_0 by one and the incoming VPs at n_m by one. For the intermediate nodes $n_1, \ldots n_{m-1}$, both the incoming and outgoing VP numbers are reduced by one each. The VPI limits are also different for the Network-to-Network Interface (NNI) interface and the User-to-Network Interface (UNI) interface. For the NNI, the VPI field has 12 bits, and the maximum VPI number is 4096. For the UNI interface, the VPI field is 8 bits, and the maximum VPI number is 256. In practice, some VPIs are reserved for signaling or other management functions, so the actual number of VPI that could be used for dynamic VP assignment is less than the maximum number. It is, however, always possible to eliminate the violated VPI constraint through merge operations if the number of VP hop counts constraint is not the concern. Indeed, this is true, since through merge operations we can obtain a network that only has VPs between two nodes connected either by a physical link or a VP. Assuming that the network has n nodes, then the maximum possible number of VPIs for an ATM switch in such a network is twice the number of all connections to this node or $2(n-1)$, a condition that can be satisfied by all practical networks. The cost for this extreme case is the longer connection-setup

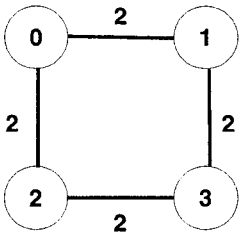
time, since the network is configured as a circuit-switched network.

4 Results

The algorithm proposed in Section 3 is implemented in C++. In our test runs, the topological data of the physical network for virtual path assignments are given. In this section, we present the results of several test cases. To demonstrate the algorithm, our first test case consists of a 4 nodes. The physical network, traffic requirements, and the virtual path assignments are shown in Fig. 5. The traffic requirements are the effective bandwidths. The iteration process is listed in Table 1. The bandwidth of VPs are assigned by the amount of flow divides the flow density. Since it is a simple case, the correctness of the solution obtained can be easily verified.

Table 1: The Iteration Process

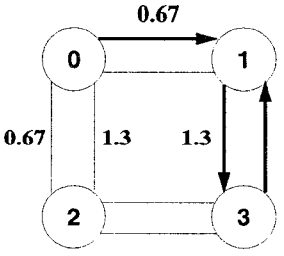
Iteration	λ_{10}	λ_{20} , λ_{32}	λ_{31}	f_{01}^{01}	$f_{02}^{01}, f_{23}^{01}, f_{31}^{01}$	f_{13}^{13}	$f_{10}^{13}, f_{02}^{13}, f_{23}^{13}$	Maximum Density
0	1	1	1	1	0	0	0	1.0
1	1.13	0.63	1.63	0.59	0.41	0.82	1.18	0.89
2	1.45	0.67	1.31	0.48	0.52	1.05	0.95	0.79
3	1.39	0.68	1.40	0.50	0.50	1.00	1.00	0.75



a) Topology with link BWs

source	dest.	BW
0	1	1
1	3	2

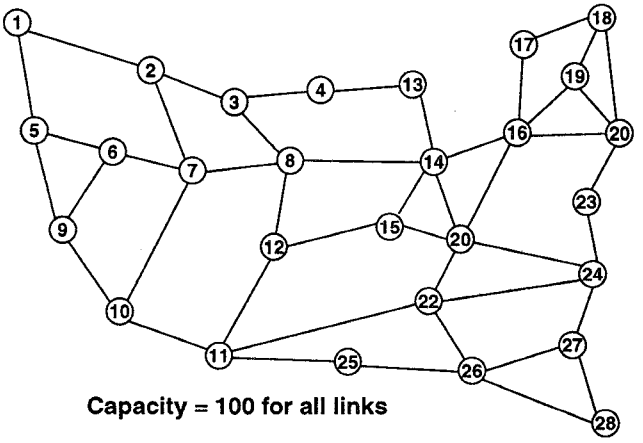
b) Traffic requirements



c) Virtual paths and BWs assigned

Fig. 5. A 4-node Test Case

The second test case is the US long-haul case-study network used in [10] with 28 nodes as shown in Fig. 6.



Capacity = 100 for all links

Fig. 6. The US Long-haul Case Study Sample Network

The traffic requirements are listed inTable 2 in terms of effective bandwidths. The resulting VP assignments are listed in Table 3. The iteration process is shown Fig. 7 in In this case, the final maximum flow density is 0.8. Since

Table 2: Traffic Demand

Source	1	2	1	3	1	4	1	5	2	3	2	4	2	5	3	4	3	5	4	5
Destination	2	1	3	1	4	1	5	1	3	2	4	2	5	2	4	3	5	3	5	4
Demand	10	10	10	10	20	20	20	20	10	10	20	20	20	20	20	20	20	20	20	20

we would like to utilize all the link capacities for virtual path assignment, the VPs are assigned the bandwidths of the flows, which are obtained from the solution, times $1/(flow\ density)$. Thus, if the flow densities are less than 1, the bandwidths assigned are larger than the amount of the flows. If the densities are greater than 1, then the VP bandwidths will be less than the flow amounts.

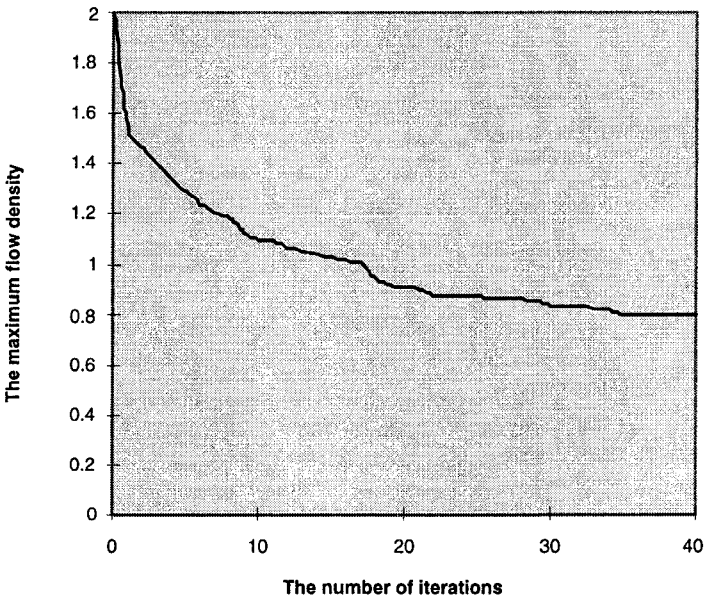


Fig. 7. The Maximum Flow Density vs. the Number of Iterations

Table 3: Traffic Flows

Source	Destination	Flow amount/ VP bandwidths	Route
1	2	10 / 14	1-2
2	1	10 / 14	2-1
1	3	10 / 12	1-2-3
3	1	10 / 12	3-2-1

Table 3: Traffic Flows

4	1	10 / 14 10 / 14	4-8-7-2-1 4-13-14-15- 12-11-10-9- 5-1
1	5	20 / 28	1-5
5	1	20 / 28	5-1
2	3	10 / 12	2-3
3	2	10 / 12	3-2
2	4	10 / 12 10 / 14	2-3-4 2-7-8-4
4	2	10 / 12 10 /14	4-3-2 4-8-7-2
2	5	10 /14 10 / 14	2-1-5 2-7-6-5
5	2	10 / 14 10 / 14	5-1-2 5-6-7-2
3	4	20 / 25	3-4
4	3	20 / 25	4-3
3	5	10 / 12 10 / 12	3-2-7-6-5 3-4-13-14- 15-12-11-10- 9-5
5	3	10 / 12 10 / 14	5-6-7-2-3 5-9-10-11- 12-15-14-13- 4-3
4	5	20 / 28	4-13-14-15- 12-11-10-9-5
5	4	20 / 28	5-9-10-11- 12-15-14-13- 4

5 Conclusions and Future Work

We propose a model for optimal virtual path assignments and bandwidth allocations. The model is formulated as a multicommodity flow problem and solved using the decomposition/coordination method. This algorithm is implemented using a linear programming solver, and the experimental results based on our test cases show that the algorithm is quite efficient. A VP merge algorithm is proposed to deal with the VPI constraints on ATM switches and cross-connects. ATM network VP management, however, involves many more aspects than what we have investigated. Issues for future studies include the protocol design for dynamic management of virtual paths, which should guarantee a seamless transition for VPs from old assignments to new assignments.

6 REFERENCES

[1] A.I. Elwalid and D.E. Mitra, "Effective Bandwidth of Bursty, Variable Rate Sources for Admission Control to B-ISDN," *ICC '93*, pp. 1325-30, vol.3, 1993.

[2] S. Ahn, R.P. Tsang, S.-R. Tong, and D.H.C. Du, "Virtual Path Layout Design on ATM Networks," *Proc. of IEEE INFOCOM '94,* Vol. 1, pp. 192-200, 1994.

[3] I. Chlamtac, A. Farago, and T. Zhang, "Optimizing the system of virtual paths," *IEEE/ACM Transactions on Networking*, vol.2, (no.6), 581-587, Dec. 1994.

[4] J. Burgin and D. Dorman, "Broadband ISDN resource Management: The Role of Virtual Paths," *IEEE Comm. Magazine*, Vol. 29 (no. 9), pp. 44-48, 1991.

[5] Z. Qin and F. F. Wu, "The Optimal Virtual Path Design on ATM Networks", Workshop on ATM Broadband Communications, The Chinese University of Hong Kong, Dec. 1995.

[6] F. Lin and K. Cheng, "Virtual Path Assignment and Virtual Circuit Routing in ATM Networks," *Proc. of Globecom'93,* November, Vol. 1, pp 436-41, 1994.

[7] K. Cheng and F. Lin, "On the Joint Virtual Path Assignment and Virtual Circuit Routing Problem in ATM Networks," *Proc. of Globecom'94,* Vol. 2, pp. 777-782, 1994.

[8] C. Lawrence, J. L. Zhou, and A. L. Tits, "User's Guide for CFSQP Version 2.3: A Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints," Electrical Engineering Department and Ins. for Systems Research, University of Maryland, 1995.

[9] M. Logothetis and S. Shioda, "Centralized Virtual Path Bandwidth Allocation Scheme for ATM Networks," *IEICE Trans. Commun.,* vol. E75-B. No. 10, pp. 1071-1080, October, 1992.

[10] K. Murakami and H. S. Kim, "Near-Optimal Virtual Path Routing for Survivable ATM Network," *Proc. of Inforcom'94,* Vol. 1, pp. 208-215, 1994.

[11] Z. Qin, F. Wu, "Designing B-ISDN Network Topologies Using the Genetic Algorithm," *MASCOTS*, 1997, pp. 140-145, Israel.

[12] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, "Network Flows: Theory, Algorithms, and Applications," *Prentice Hall, Inc.*, 1993.