Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 158 (2004) 555–569
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

ATM VP-based network design

Jangha Kang [a], Kyungchul Park [b], Sungsoo Park [a,*]

[a] *Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-Dong, Yuseong-Gu, Taejon 305-701, South Korea*
[b] *Telecommunication Network Lab, Korea Telecom, Taejon 305-390, South Korea*

Received 24 October 2001; accepted 16 January 2003
Available online 12 August 2003

Abstract

We consider the problem of designing an ATM VP-based leased line backbone network. Given point-to-point communication demands having predefined sizes in a network, the problem is to find configurations of demand routes and link facilities installed on each edge satisfying all demands at minimum cost under some constraints. One of the most important constraints is that a single demand cannot be split over multiple link facilities. This is a sort of bin packing constraint. We propose an integer programming formulation of the problem and an algorithm to solve it. An efficient column generation technique to solve the linear programming relaxation is proposed, and a valid inequality is used to strengthen the integer programming formulation. The algorithm incorporates the column generation technique and the cutting plane approach into a branch-and-bound scheme.

We test the proposed algorithm on some real problems. The results show that the algorithm can be used to solve the problems within reasonably small computing times.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Integer programming; Column generation; Cutting plane; Bin packing; Routing; Telecommunications

1. Introduction

The emerging information networking services demand a large bandwidth from telecommunication networks for large volumes of data to be exchanged within a very short time interval. Transport technologies to meet these broadband communications needs include asynchronous transfer mode (ATM). ATM is a high-speed, integrated multiplexing and switching technology that transmits information across telecommunications and computer networks using the data transmission rate required by the customer through fixed-length cells that are connection-oriented [14].

The ATM is capable of supporting a wide variety of connections with different bandwidth requirements and traffic characteristics by means of resource management control. As an effective way to facilitate the coexistence of traffic with diverse traffic characteristics and different quality of service (QOS) requirements in ATM networks, a

* Corresponding author. Tel.: +82-428693121; fax: +82-428693110.
E-mail address: sspark@kaist.ac.kr (S. Park).

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0377-2217(03)00372-2

virtual path (VP) concept has been proposed. A VP is defined as a direct logical connection between two end nodes to accommodate multiple virtual channels (VCs) simultaneously. VPs are multiplexed on the physical transmission link in a cell-multiplexing manner. When a VP is established between two nodes, all VCs on that VP are also multiplexed and routed via an identical predefined path [13]. A VP has a predefined capacity, referred to as a bandwidth. Especially, a VP, in some leased line networks, has a bi-directional bandwidth selected from a small set of alternatives. For example, there are seven alternatives, namely 64 Kbps, 128 Kbps, 256 Kbps, 512 Kbps, 1.024 Mbps, 2.048 Mbps and 45.056 Mbps in a network based on synchronous digital hierarchy (SDH) transmission system. Note that the smaller bandwidths, in the previous example, divide the larger ones exactly. Bandwidths are called divisible in this case.

ATM usually consists of two hierarchical network layers: the higher backbone network layer and the lower local network layer. The backbone network is composed of edge nodes and core nodes. Each edge node is a hub of a cluster which includes several access nodes, where a cluster is a collection of access nodes that are served by the same edge node. On each edge node, VPs of inter cluster and intra cluster traffic are cross-connected. On a core node, just the VPs of inter cluster traffic are cross-connected. Due to the high traffic requirements and reliability requirements in the backbone network, connectivity between nodes is usually high, that is, mesh topology is usually used for the backbone network. The local network is composed of an edge node and access nodes in a cluster. Access nodes accommodate subscriber's demand (VPs). In a local network, several topologies can be used: tree, star, ring, etc. When the tree topology is used, multiplexers (MUX) are installed on some of the access nodes to switch the VPs of descendant access nodes in the tree.

On the edge node and core node, virtual path cross-connect systems (VPXs) are installed. VPXs are connected by link facilities, each of which forms a transmission path. The link facilities are selected from a small set of alternatives. For example, in synchronous digital hierarchy (SDH) networks, there are the following high-speed transmission rates of link facilities, referred to as link capacities: STM-1 (155.52 Mbps), STM-4 (622.08 Mbps), STM-16 (2488.32 Mbps) and STM-64 (9953.28 Mbps), where STM means synchronous transfer module [14]. Note that the smaller transmission rates divide the larger ones exactly. In this case, we say that the rates or the link capacities are divisible. These transmission rates are bi-directional, i.e. a link facility of STM-1 can accommodate 155.52 Mbps in both directions simultaneously. An example of an ATM network is illustrated in Fig. 1.

A VPX has a given number of ports on which a link facility is installed. A link facility uses a predefined number of ports exclusively. For example, a link facility of STM-1 uses one port and a link facility of STM-4 uses four ports. Hence, if a set of link facilities requires more than the available number of ports on each VPX, they cannot be installed on it simultaneously. We call this "port constraint".

A number of VPs established between a pair of VPXs may be divided and loaded on different paths. However, a single VP cannot be split and pass through different physical link facilities, that is, a single VP should be loaded on a single link facility between two adjacent VPXs. This comes from an ITU-T (Telecommunication Standardization Sector of International Telecommunication Union) recommendation saying that "a VP link must be within a single transmission path (TP), and thus cannot be split over multiple TPs" [5]. We call this "bin packing constraint".

One fundamental problem arises in designing an ATM VP-based leased line backbone network. We are given a graph $G = (V, E)$, where node set V is the set of VPXs and edge set E is the set of possible connections between VPXs, the number of available ports for each VPX, a set of link facilities with installation costs for each edge, and a set of demands whose bandwidths are selected from a small set of alternatives. The problem is to find a configuration of VP routes and link facilities to be installed on each edge to carry the demands at minimum cost under the constraints that the number of ports used should be less than or equal to the given number for each node (the port con-

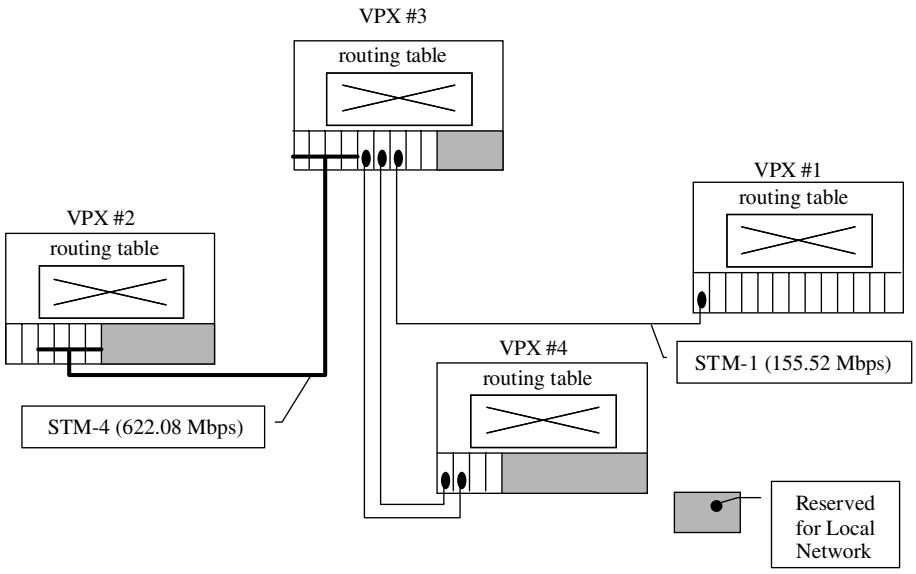

Fig. 1. An example of an ATM network.

straint) and that a single VP cannot be split (the bin packing constraint). We refer to the problem as the ATM VP-based network design problem (VPNDP).

Theorem 1. *VPNDP is NP-hard.*

Proof. See Appendix A.1. □

There are two related problems that have been studied in the literature: the bandwidth packing problem and the capacitated network design problem. Both of them are NP-hard. Given edge capacities and edge costs, the bandwidth packing problem is to find a configuration of demand routing at minimum cost. In this problem, the capacities of edges are fixed and each demand cannot be split. Laguna and Glover [7] used the Tabu search method, and Park et al. [12] used some integer programming approaches to solve it.

The capacitated network design problem is to find a configuration of demand routing and facilities installed on each edge at minimum facility cost. In this problem, the capacity of an edge is equal to the sum of the transmission rates of the link facilities installed on the edge such that the demands are allowed to be split by fractional values. To solve this problem, Magnanti et al. [8] used integer programming approaches assuming that only two types of link facilities are available.

In this paper, we propose an integer programming formulation of VPNDP and an algorithm to solve it. An efficient column generation technique to solve the linear programming (LP) relaxation is proposed, and a class of valid inequalities is characterized, so that the inequalities are appended to the integer model to strengthen the LP relaxation. The column generation technique and the cutting plane approach are embedded in a branch-and-bound scheme. For details of integer programming and column generation techniques, refer to [1,11,12].

VPNDP is more difficult to solve than the capacitated network design problem due to the bin packing constraint. We first give an efficient representation of the bin packing constraint by a system of linear inequalities and integrality restrictions.

The paper is organized as follows. In the following section, we give the details of the representations of the bin packing constraint. Section 3 presents the formulation of VPNDP. Section 4 gives the column generation procedure to solve the LP relaxation. In Section 5, a valid inequality is

introduced to strengthen the initial formulation. Section 6 describes the basic steps of the proposed algorithm. Computational results are reported in Section 7. Finally, Section 8 concludes.

2. Bin packing constraint

Before presenting the representations of the bin packing constraint, let us introduce the bin packing problem. It is, in general, a combinatorial optimization problem involving the partition of a set of items into subsets. Given a set of bins with size b and a set of items $T = \{1, 2, \ldots, n\}$ with a size a_t for $t \in T$, the problem is to assign each item to one bin so that the total size of the items in each bin does not exceed b and the number of used bins is minimized. Since the problem is NP-hard in the strong sense [4] (it contains 3-PARTITION as a special case), a number of simple approximation algorithms for the problem have been found. One of the approximation algorithms which is well known is the "first-fit decreasing" (FFD) algorithm. When the items are sorted as $a_1 \geqslant a_2 \geqslant \cdots \geqslant a_n$, FFD considers them according to their order. In the first step, the algorithm initializes a bin with index 1. Then it assigns each item to the lowest indexed initialized bin into which it fits; only when the current item cannot fit into any initialized bin, a new bin is introduced with index $j+1$, where j is the number of initialized bins at hand. It is worth to note that FFD gives an optimal solution when $a_{t+1}|a_t$ for each $t = 1, \ldots, n-1$, that is, a_{t+1} exactly divides a_t for $t = 1, \ldots, n-1$ [2].

A generalization of the bin packing problem is the "variable sized bin packing problem". It considers a set of bins F with size b_f and cost c_f for $f \in F = \{1, 2, \ldots, |F|\}$. The objective is to minimize the total cost of the bins to use. Elsewhere [6], we have proposed two approximation algorithms, namely, "iterative first-fit decreasing" (IFFD) and "iterative best-fit decreasing" (IBFD) and shown that they give optimal solutions when $a_n|\cdots|a_1$, $b_{|F|}|\cdots|b_1$, and $c_1/b_1 \leqslant c_2/b_2 \leqslant \cdots \leqslant c_{|F|}/b_{|F|}$. In this paper, the problem will be called "bin packing problem", in short.

In the remainder of the section, two models of the bin packing problem are given via systems of linear inequalities and integrality restrictions that consider multiple items for each type, say, n_t items of size a_t for each $t \in T$.

2.1. Modeling the bin packing problem

The following is a "natural representation" of the solution set for the bin packing problem for the case with two bin types, i.e. $|F| = 2$:

$$\mathbf{P} = \left\{ (x,y) \in \mathbf{Z}_+^{2nm} \times \mathbf{B}^{2m} : \sum_{t\in T} a_t x_{it}^f \leqslant b_f y_i^f, \right.$$
$$\left. i = 1, \ldots, m, f = 1, 2, \sum_{i=1}^{m} x_{it}^1 + \sum_{i=1}^{m} x_{it}^2 = n_t, t \in T \right\},$$

where the 0–1 variable y_i^f takes the value 1 if bin i of type f is used, and, otherwise its value is 0, and the integer variable x_{it}^f gives the number of items of type t assigned to the bin i of type f. On the other hand, m is a large enough upper bound on the number of used bins of each type.

Under the assumption that $b_2|b_1$ and $a_{t+1}|a_t$ for all $t = 1, \ldots, n-1$, an equivalent representation with fewer variables is as follows:

$$\mathbf{D} = \left\{ y \in \mathbf{Z}_+^2 : \sum_{t\in T} \left\lfloor \frac{a_t}{a_k} \right\rfloor n_t \leqslant \left\lfloor \frac{b_1}{a_k} \right\rfloor y_1 \right.$$
$$\left. + \left\lfloor \frac{b_2}{a_k} \right\rfloor y_2, k \in T \right\},$$

where the integer variable y_f gives the number of used bins of type f, and $\lfloor w \rfloor$ denotes the largest integer smaller than or equal to a real value w.

The assignment of each item is not explicitly considered while defining the set **D**. However, we can easily check whether a set of items can be packed into a set of bins or not. The validity of the representation will be shown in Theorem 3.

Before showing the validity, let us recall a known result for the representation of the convex hull of solutions to the knapsack problem.

Theorem 2 [9]. *With $a_{t+1}|a_t$ for all $t = 1, \ldots, n-1$, the convex hull of $\{x \in \mathbf{Z}_+^n : \sum_{t\in T} a_t x_t \leqslant b\}$ is described by the nonnegativity constraints on the variables and the n following inequalities*:

$$\sum_{t\in T}\left\lfloor \frac{a_t}{a_k}\right\rfloor x_t \leqslant \left\lfloor \frac{b}{a_k}\right\rfloor, \quad k\in T.$$

Theorem 3. *Suppose* $b_2|b_1$ *and* $a_{t+1}|a_t$ *for all* $t=1,\ldots,n-1$. *Considering a number* n_t *of items for each* $t\in T$, *there exists an integer vector* $y\in\mathbf{D}$ *if and only if there exists an integer vector* $(\tilde{x},\tilde{y})\in\mathbf{P}$ *such that* $y_f=\sum_{i=1}^{m}\tilde{y}_i^f$ *for* $f=1,2$.

Proof. See Appendix A.2. □

An important implication from Theorem 3 is that it can be checked whether a set of items can be packed into a set of bins without needing to know the explicit assignment of items but by using the following inequalities:

$$\sum_{t\in T}\left\lfloor \frac{a_t}{a_k}\right\rfloor x_t \leqslant \left\lfloor \frac{b_1}{a_k}\right\rfloor y_1+\left\lfloor \frac{b_2}{a_k}\right\rfloor y_2, \quad k\in T, \tag{1}$$

where x_t denotes the number of items of size a_t, $t\in T$ and y_f denotes the number of used bins of size b_f, $f=1,2$, when $b_2|b_1$ and $a_{t+1}|a_t$ for all $t=1,\ldots,n-1$. If an integral vector (x,y) satisfies the inequalities (1), we can always get an explicit assignment by using the modified FFD algorithm. Notice that we can generalize the inequalities (1) for a set of bins $F=\{1,\ldots,|F|\}$ such that $|F|>2$ and $b_{f+1}|b_f$, for all $f=1,\ldots,|F|-1$, as follows:

$$\sum_{t\in T}\left\lfloor \frac{a_t}{a_k}\right\rfloor x_t \leqslant \sum_{f\in F}\left\lfloor \frac{b_f}{a_k}\right\rfloor y_f, \quad k\in T. \tag{2}$$

3. Modeling the ATM VP-based network design problem

In this section, we present the formulation of the ATM VP-based network design problem (VPNDP), which determines the number of link facilities to be installed on each edge and the routing of the virtual paths. For this purpose, in this paper it is assumed that only two types of link facilities are available (as Magnanti et al. [8]). The two facilities may correspond to STM-4 and STM-1, STM-64 and STM-16, or any typical case in which the smaller transmission rate exactly divides the larger one. In the network of interest, the following predefined VP bandwidths are considered: 45.056 Mbps, 2.048 Mbps, 1.024 Mbps, 512 Kbps, 256 Kbps, 128 Kbps and 64 Kbps. These bandwidths are divisible, so we assume that the demand sizes are divisible. Note that, even under these assumptions, the problem is still NP-hard (see Appendix A.1 for details).

First, let us introduce some notation to be used in the formulation.

- $G=(V,E)$ undirected graph, where V is the set of vertices representing the VPXs and E is the set of edges representing the potential connections between VPXs.
- $\delta(v)$ set of edges incident to node v for $v\in V$.
- T set of demand types.
- P set of node pairs where demands are defined.
- R_p^t set of VP routes where the demand of type t between the node pair p may be loaded for $t\in T$, $p\in P$. Note that two VP routes, say, $r_1\in R_p^{t_1}$ and $r_2\in R_p^{t_2}$, $t_1\neq t_2$, can use the same physical path.
- $R_p^t(e)$ set of VP routes in R_p^t that include edge e for $e\in E$.
- c_f^e installation cost of link facility type f on edge e for $f=1,2$, $e\in E$.
- b_f size (capacity) of link facility f for $f=1,2$, where $b_1\geqslant b_2$ and $b_2|b_1$.
- a_t size (bandwidth) of demand type t for $t\in T$, where $a_1\geqslant a_2\geqslant\cdots\geqslant a_{|T|}$ and $a_{|T|}|\cdots|a_1$.
- τ_f number of ports that are required by link facility f for $f=1,2$.
- d_v number of available ports for node v for $v\in V$.
- n_p^t number of demands (VPs) of type t between the node pair p for $t\in T$, $p\in P$.

The variables of the problem are as follows:

- y_f^e integer variable that gives the number of link facilities of type f installed on edge e for $f=1,2$, $e\in E$.
- x_r integer variable that gives the number of the loaded demands on the route r, where r is a VP route to load the demands

of type t between a node pair p for $r \in R_p^t$, $t \in T$, $p \in P$.

The pure integer programming model is as follows:

$$(\text{MP}) \quad \min \quad \sum_{e\in E}\sum_{f=1}^{2} c_f^e y_f^e \tag{3}$$

$$\text{s.t.} \quad \sum_{r\in R_p^t} x_r \geqslant n_p^t \quad \forall t \in T,\ p \in P, \tag{4}$$

$$\sum_{e\in\delta(v)}\sum_{f=1}^{2} \tau_f y_f^e \leqslant d_v \quad \forall v \in V, \tag{5}$$

$$\sum_{f=1}^{2} \left\lfloor \frac{b_f}{a_k} \right\rfloor y_f^e - \sum_{t\in T}\sum_{p\in P}\sum_{r\in R_p^t(e)} \left\lfloor \frac{a_t}{a_k} \right\rfloor x_r \geqslant 0 \quad \forall e \in E,\ k \in T, \tag{6}$$

$$x_r \in \mathbf{Z}_+ \quad \forall r \in \bigcup_{t\in T}\bigcup_{p\in P} R_p^t \quad \text{and}$$

$$y_f^e \in \mathbf{Z}_+ \quad \forall e \in E,\ f = 1,2. \tag{7}$$

Without loss of generality, we can assume that all data are integers. Constraints (4) ensure that the demand should be satisfied. Constraints (5) bound the number of ports to be used at each node. The constraints (6) jointly with the integrality constraints (7) ensure that each VP will not be split over multiple link facilities, notice that it is the bin packing constraint.

It is easy to see that the validity of MP requires that the bandwidths of VPs and the capacities of the link facilities are divisible, i.e. $b_2|b_1$ and $a_n|\cdots|a_1$.

A solution to MP gives the VPs of each type t passing through an edge e, which is equal to $\sum_{p\in P}\sum_{r\in R_p^t(e)} x_r$, and the number of link facilities of each type f installed on edge e, which is equal to y_f^e.

A configuration of VPs loaded on each link facilities can be found by using the modified FFD for each edge e where the number of items n_t is $\sum_{p\in P}\sum_{r\in R_p^t(e)} x_r$ for each type t and the number of installed bins y_f is y_f^e for each type f. For details of the algorithm, see Appendix A.2. The algorithm will be introduced in the proof of Theorem 3.

Anyway, we can see that MP has exponentially many variables. However, we can solve the LP relaxation efficiently by the (delayed) column generation technique. It has been successfully used for solving the bandwidth packing problem and many other hard combinatorial optimization problems, see [1,12].

4. Column generation to MP

Let (LP) be the LP relaxation of MP. Now we will describe the column generation procedure to solve LP. In the procedure, we construct a restricted LP relaxation, referred to as RLP, which has all the y-variables and a subset of VP routes at the beginning of the procedure and other VP routes are appended when they are needed. Let α_{pt} be the dual variable associated with the constraint (4) for node pair $p \in P$ and demand type $t \in T$, γ_{ek} be the dual variable associated with the constraint (6) for edge $e \in E$ and demand type $k \in T$, and α_{pt}^* and γ_{ek}^* be the values of the optimal dual solution of the current RLP. Finally, let L_r be the set of edges that belong to the VP route r. Then, the optimal solution of current RLP is optimal to LP, provided that

$$\sum_{e\in L_r}\left(\sum_{k\in T}\gamma_{ek}^*\left\lfloor\frac{a_t}{a_k}\right\rfloor\right) \geqslant \alpha_{pt}^* \quad \forall r \in R_p^t, t \in T, \text{ and } p \in P,$$

and the corresponding reduced cost for y_f^e is nonnegative for each $e \in E$ and $f = 1,2$. So, the column generation problem associated with node pair $p \in P$ and demand type $t \in T$ can be expressed as follows:

$$(\text{SP}(p,t)) \quad \min \quad \sum_{e\in L_r}\left(\sum_{k\in T}\gamma_{ek}^*\left\lfloor\frac{a_t}{a_k}\right\rfloor\right)$$
$$\text{s.t.} \quad r \in R_p^t,$$

where $\text{SP}(p,t)$ is the problem that finds the shortest path between node pair p with edge weight $\sum_{k\in T}\gamma_{ek}^*\lfloor a_t/a_k\rfloor$ for each edge e. Since the edge weights are nonnegative, $\text{SP}(p,t)$ can be solved efficiently by the Dijkstra's algorithm [11]. If the length of the shortest path is less than α_{pt}^*, the path

can be appended to the current formulation. Otherwise, no column is generated with respect to node pair p and demand type t. If the length of the shortest paths are greater than or equal to α^*_{pt} for all $p \in P$ and $t \in T$, then the current solution is optimal to LP.

5. Cutting planes

Consider the following notation that is required for presenting a valid inequality that can be used to cut off fractional solutions in LP. The node set V is partitioned into the two node sets S and $\overline{S}$, $E(S,\overline{S}) = \{(i,j) \in E : i \in S, j \in \overline{S} \text{ or } i \in \overline{S}, j \in S\}$, and $P(S,\overline{S}) = \{(s,t) \in P : s \in S, t \in \overline{S} \text{ or } s \in \overline{S}, t \in S\}$. Let also $y_f(S) = \sum_{e \in E(S,\overline{S})} y^e_f$ for each $f \in F$, and $n_t(S) = \sum_{p \in P(S,\overline{S})} n^t_p$ for each $t \in T$. So, it is easy to see that the following two inequalities are valid for MP:

$$b_1 y_1(S) + b_2 y_2(S) \geqslant \sum_{t \in T} a_t n_t(S)$$

and

$$\frac{b_1}{b_2} y_1(S) + y_2(S) \geqslant \left\lceil \frac{\sum_{t \in T} a_t n_t(S)}{b_2} \right\rceil. \tag{8}$$

By letting $C = b_1/b_2$ and $D(S) = \lceil \sum_{t \in T} a_t n_t(S)/b_2 \rceil$, we can rewrite (8) as follows:

$$C y_1(S) + y_2(S) \geqslant D(S).$$

By using (8), Magnanti et al. [8] proposed the following inequality:

$$r y_1(S) + y_2(S) \geqslant r \left\lceil \frac{D(S)}{C} \right\rceil, \tag{9}$$

where $r \equiv D(S) \bmod (C)$. It is valid for the two facility-loading problem (TFLP). In fact, it defines a facet of the convex hull of the solutions to TFLP if and only if $D(S) > 0$ and the both sub-graphs defined by S and $\overline{S}$ are connected. Note that (9) is also valid for VPNDP.

A column generation approach embedded in the branch-and-bound procedure has been applied successfully to solve many large-scale mixed integer programming problems. The approach is referred to as the branch-and-price approach. The branch-and-price approach, if combined with strong cutting planes, can become more efficient. Within our knowledge there exist very few papers addressing this approach, because it is hard to design a column generation scheme if a cutting plane has variables that need to be generated by a column generation procedure. Some applications can be found in [10,12].

However, in our case, there is no path variable in the inequality (9). And so, the column generation procedure is not affected even after some cutting planes are appended to LP.

6. Algorithmic framework

6.1. Overview

In this section, we give a brief presentation of the proposed algorithm for solving the network design problem. To start the column generation procedure, an initial RLP must be provided. This initial RLP should be feasible to ensure that the proper dual information is passed to the column generation problem. With the columns related to the link facilities (y's), a dummy column consisting of 1's in the rows corresponding the inequalities (4), -1's in the rows corresponding to the inequalities (5), 0's in the other rows and a large cost in the objective function, the initial RLP can be constructed by using the shortest path for each node pair $p \in P$ and demand type $t \in T$. The shortest paths are generated by the Dijkstra's algorithm, considering c^e_1 as the cost of each edge e. The dummy column ensures that a feasible solution to the RLP exists. This dummy column will be kept in RLP at each node of the branch-and-bound tree for the same reason.

If the optimal solution to the initial RLP is not dual feasible, then new columns are generated and appended to RLP. Otherwise, no more columns need to be appended to RLP. And so, the procedure for identifying cutting planes violated by the current solution is invoked such that the inequalities are appended to RLP.

After appending the inequalities, we go through the same procedure as we do after the initial RLP is constructed. If the solution of RLP is dual infeasible, the required columns are generated and

appended to RLP until it is optimally solved. When no more cutting planes can be found and, hence, no more columns can be generated, if the solution obtained by solving the last RLP is integer, it is an optimal solution to MP. Otherwise, an optimal integer solution is obtained by using a branch-and-cut procedure that includes the cutting plane approach with column generation. The overall algorithm is presented in Fig. 2.

6.2. Cutting plane identification

The cutting planes to identify are from the type of valid inequality (9). It is hard to find an inequality (9) that is violated by the current optimal solution to RLP due to the structure of the inequality; we simply use a total enumeration scheme. Since each inequality (9) is defined for a node partition, we generate all node partitions to check whether any violated inequality exists. Our preliminary experience has generated a few violated inequalities. So, we only append the first generated inequality per iteration. Magnanti et al. [8] present a heuristic for identifying violated inequalities which is based on a partial enumeration scheme, because the networks that are considered are larger than ours.

6.3. Branch-and-cut

To force the integrality of variables, we perform the branch-and-cut procedure by solving RLP at each node of the branch-and-bound tree such that the cutting plane and column generation schemes are used.

Given an optimal solution $(\tilde{y}, \tilde{x})$ to RLP at a node of the branch-and-bound tree, the branching variable selection rule is as follows: y-variables have priority over the x-variables; the y-variable whose current value, say, $\tilde{y}_i^e$ is closest to $\lfloor \tilde{y}_i^e \rfloor + 0.5$ is branched first; the x-variables have the same branching rule as the y-variables.

For speeding up the branch-and-cut process, a heuristic is executed at each branch-and-bound node where the y-variables take integer values in

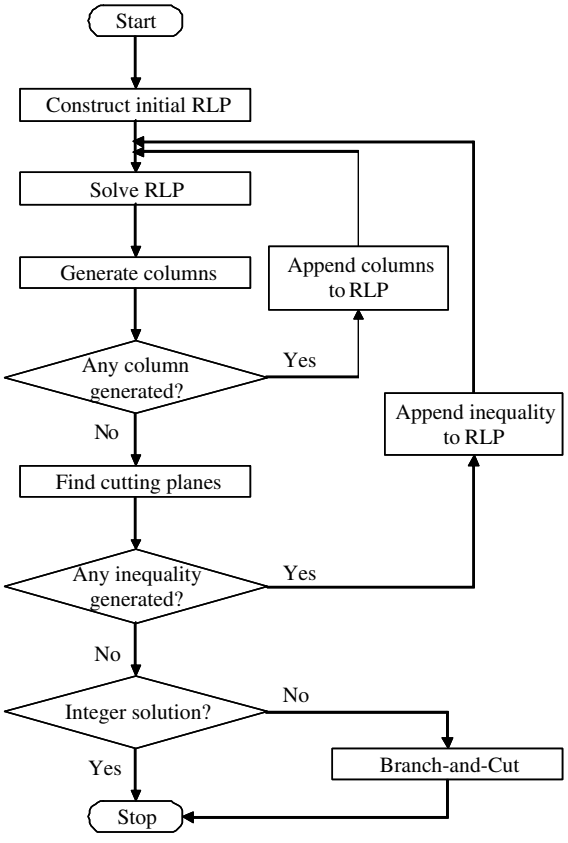

Fig. 2. Diagram of the algorithm.

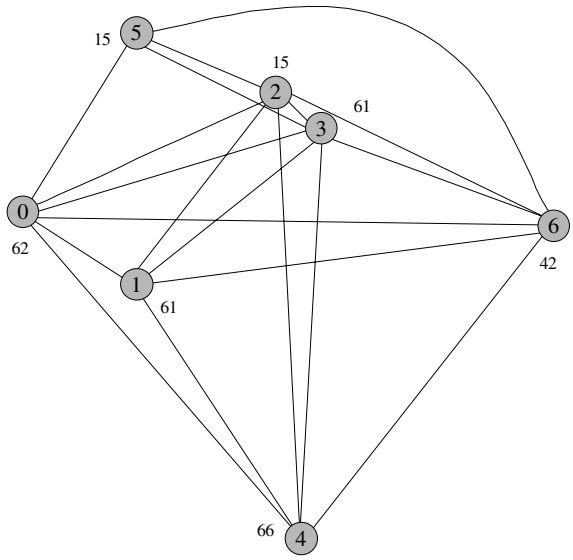

Fig. 3. Network 1 ($|V| = 7, |E| = 19$).

the optimal solution to the RLP. The heuristic fixes the y-variables to its integer value ($\tilde{y}$) and solves the reduced problem by using the branch-and-bound method up to optimality such that the optimal solution, if any, is feasible for the original problem and can replace the incumbent solution, in case. In our computational study, we observed that this procedure dramatically reduces the computation time in all cases.

Before the branch-and-bound procedure is used, finding shortest paths is sufficient for column generation. However, the shortest path column generation for some branch-and-bound node can make again the columns related to a path variable once it has been bounded above. To overcome this difficulty, the k-shortest path algorithm due to Yen [15] is used instead of the shortest path algorithm, such that in case the generated column already

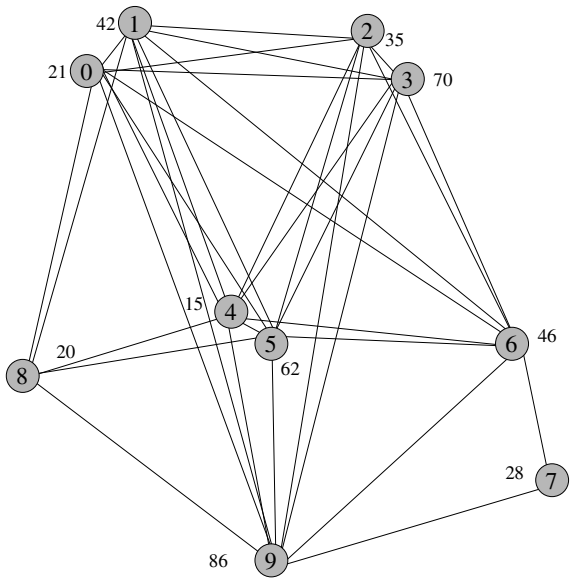

Fig. 4. Network 2 ($|V| = 10, |E| = 35$).

Table 1
Installation costs for network 1 ($|V| = 7, |E| = 19$)

e	From	To	c_1^e	c_2^e
0	0	1	150	59
1	0	2	300	127
2	0	3	300	127
3	0	4	450	205
4	0	5	250	115
5	0	6	600	267
6	1	2	250	105
7	1	3	250	105
8	1	4	350	155
9	1	6	500	223
10	2	3	50	20
11	2	4	400	178
12	2	5	150	72
13	2	6	300	146
14	3	4	400	178
15	3	5	150	72
16	3	6	300	146
17	4	6	150	70
18	5	6	450	211

Table 2
Installation costs for network 2 ($|V| = 10, |E| = 35$)

e	From	To	c_1^e	c_2^e
0	0	1	250	100
1	0	2	400	159
2	0	3	400	159
3	0	4	570	227
4	0	5	570	227
5	0	6	700	305
6	0	8	500	215
7	0	9	800	367
8	1	2	350	159
9	1	3	350	159
10	1	4	500	227
11	1	5	500	227
12	1	6	750	305
13	1	8	500	215
14	1	9	800	367
15	2	3	250	100
16	2	4	450	205
17	2	5	450	205
18	2	6	550	255
19	2	9	700	323
20	3	4	450	205
21	3	5	450	205
22	3	6	550	255
23	3	9	700	323
24	4	5	250	100
25	4	6	600	278
26	4	8	350	172
27	4	9	650	246
28	5	6	650	278
29	5	8	350	172
30	5	9	550	246
31	6	7	400	176
32	6	9	400	170
33	7	9	500	208
34	8	9	650	311

exists, a new one in the shortest path ordering is checked. If it does not exist, the column is generated, see also Park et al. [12].

7. Computational results

The proposed approach has been tested by using the networks shown in Figs. 3 and 4, where the numbers of available ports at the nodes are shown beside the circles. Tables 1 and 2 show the installation costs of link facilities for each network. Tables 3 and 4 give the corresponding flow demand. The networks are real ATM networks under consideration in Korea, but data has been perturbed to preserve confidentiality. Two types of link facilities are considered, say, 622.08 and 155.52 Mbps; therefore, 4 ports and 1 port are required, respectively.

The obtained optimal solutions for the two networks are given in Tables 5 and 6 and their total installation costs are 5120 and 7664, respectively. Both tables only show the numbers of installed link facilities on the edges (y's).

For enlarging the computational experimentation, new instances for each network topology were created by modifying the real cases.

The problem dimensions are shown in Tables 7 and 8. In both tables, the instances 10 are the real cases. The headings are as follows: Set of VPs, where B1: 45.056 Mbps, 2.048 Mbps, 1.024 Mbps, 512 Kbps, 256 Kbps, 128 Kbps and 64 Kbps, and B2: 45.056 Mbps, 2.048 Mbps and 1.024 Mbps; nVPs, the total number of VPs; TB: the total bandwidth of the demand in Giga-bits.

The algorithm was coded in C, and the CPLEX 7.0 callable library routines were used as LP solver. The computational experimentation was performed on a Pentium 500 MHz PC with 256 MB RAM. Tables 9 and 10 summarize the computational results. The headings are as follows: Z_{LP0}, the optimal value of LP relaxation without the inequalities (9); Z_{LP}, the optimal value of LP relaxation with the inequalities (9); Z_{IP}, the value of the optimal integer solution; $\text{Gap0} = (Z_{IP} - Z_{LP0})/Z_{IP} \times 100$; $\text{Gap} = (Z_{IP} - Z_{LP})/Z_{IP} \times 100$; m, the number of constraints in the initial RLP; n, the number of variables in the initial RLP; nCuts,

Table 3
Flow demands for network 1 ($|V| = 7, |E| = 19$)

No.	From	To	Bandwidth (Kbps)						
			45,056	2048	1024	512	256	128	64
0	0	1	6	290	287	26	50	189	826
1	0	2	2	29	21	2	10	30	139
2	0	3	3	95	171	22	36	184	569
3	0	4	13	308	124	5	39	102	477
4	0	5	1	7	76	1	12	161	117
5	0	6	1	26	15	5	4	3	49
6	1	2	2	25	32	5	16	41	201
7	1	3	8	125	275	57	77	184	940
8	1	4	1	282	208	25	71	195	977
9	1	5	0	23	32	4	9	122	177
10	1	6	7	21	33	1	10	16	151
11	2	3	0	116	51	18	24	95	329
12	2	4	0	158	69	9	20	51	162
13	2	5	0	28	10	2	3	30	90
14	2	6	0	36	11	0	2	80	42
15	3	4	0	257	84	49	83	264	749
16	3	5	0	71	35	5	12	138	219
17	3	6	5	91	12	6	10	149	102
18	4	5	0	110	24	1	13	154	164
19	4	6	0	138	85	7	10	54	121
20	5	6	0	39	4	0	4	63	19

Table 4
Flow demands for network 2 ($|V| = 10, |E| = 35$)

No.	From	To	Bandwidth (Kbps)						
			45,056	2048	1024	512	256	128	64
0	0	3	1	74	102	0	9	39	247
1	0	5	1	7	8	0	4	19	46
2	0	6	2	5	19	0	3	16	33
3	0	9	0	70	0	0	2	2	10
4	1	3	0	14	21	0	1	6	30
5	1	4	0	2	21	0	0	2	18
6	1	5	0	1	60	0	2	10	24
7	1	9	0	35	6	0	0	0	12
8	2	3	0	105	147	16	29	109	558
9	2	5	9	67	168	30	47	187	716
10	2	6	0	28	63	9	42	106	304
11	2	8	0	10	17	2	6	112	109
12	2	9	3	172	17	0	3	16	159
13	3	4	3	69	67	10	27	53	362
14	3	5	3	107	167	29	50	167	688
15	3	6	1	26	132	8	43	108	677
16	3	8	0	10	16	2	12	129	143
17	3	9	4	198	44	10	23	35	239
18	4	5	0	100	72	39	38	140	434
19	4	6	2	13	35	11	34	62	119
20	5	6	0	83	37	33	54	165	603
21	5	8	0	61	19	6	7	129	207
22	5	9	5	315	60	6	17	240	204
23	6	9	0	117	64	2	19	54	187
24	8	9	0	136	1	0	5	109	37

Table 5
The obtained solution for network 1 ($|V| = 7, |E| = 19$)

e	From	To	y_1^e	y_2^e
0	0	1	3	0
1	0	2	0	0
2	0	3	1	0
3	0	4	2	0
4	0	5	1	0
5	0	6	0	0
6	1	2	1	0
7	1	3	2	0
8	1	4	2	0
9	1	6	0	0
10	2	3	1	1
11	2	4	1	0
12	2	5	0	0
13	2	6	0	0
14	3	4	1	0
15	3	5	1	0
16	3	6	2	0
17	4	6	1	0
18	5	6	0	0

the number of appended cuts (9); tc, total number of columns in the model at the end of the algorithms execution; nit, the number of calls to LP solver; nn, the number of the branch-and-bound nodes; time, total CPU time (second).

We can see in both tables the LP bound is dramatically improved by appending the inequalities (9). On the other hand, as $|E|$, $|T|$ and $|P|$ increase, the algorithm usually takes more time. The Gap is relatively small (within 10%) and an optimal solution is obtained within 4 hours for any instance. Since the network 2 is a large real network, we can say that the algorithm performs well for real data.

8. Concluding remarks

In this paper, we have proposed the integer programming formulation of the ATM VP-based

Table 6
The obtained solution for network 2 ($|V| = 10, |E| = 35$)

e	From	To	y_1^e	y_2^e
0	0	1	0	1
1	0	2	0	1
2	0	3	1	0
3	0	4	0	0
4	0	5	0	0
5	0	6	0	0
6	0	8	0	0
7	0	9	0	0
8	1	2	0	0
9	1	3	0	0
10	1	4	0	0
11	1	5	0	0
12	1	6	0	1
13	1	8	0	0
14	1	9	0	0
15	2	3	0	2
16	2	4	0	0
17	2	5	2	
18	2	6	0	0
19	2	9	1	0
20	3	4	1	0
21	3	5	1	0
22	3	6	1	0
23	3	9	1	0
24	4	5	1	0
25	4	6	0	0
26	4	8	0	0
27	4	9	0	0
28	5	6	1	0
29	5	8	1	0
30	5	9	2	0
31	6	7	0	0
32	6	9	1	0
33	7	9	0	0
34	8	9	0	0

Table 7
Problem instances for network 1 ($|V| = 7, |E| = 19$)

Instance	$\|P\|$	Set of VPs	nVPs	TB
1	10	B2	3002	6.384640
2	13	B2	3561	7.435264
3	16	B2	3758	8.135680
4	19	B2	3898	8.413184
5	21	B2	3983	8.565760
6	10	B1	10,596	7.152320
7	13	B1	11,893	8.277504
8	16	B1	12,740	9.040704
9	19	B1	13,339	9.366912
10	21	B1	13,673	9.544320

Table 8
Problem instances for network 2 ($|V| = 10, |E| = 35$)

Instance	$\|P\|$	Set of VPs	nVPs	TB
1	15	B2	2707	5.830656
2	18	B2	3002	6.339584
3	21	B2	3142	6.537216
4	23	B2	3059	6.348800
5	25	B2	3222	6.665216
6	15	B1	8437	6.401920
7	18	B1	10,492	7.079936
8	21	B1	11,184	7.336960
9	23	B1	11,493	7.185152
10	25	B1	12,093	7.548928

network design problem (VPNDP) so-called MP and an efficient algorithm for problem solving. In the formulation MP, the inequalities (6) ensure that each VP is not split over multiple link facilities without actually knowing how many demands of each type are loaded on each link facility. The divisibility of the bandwidths of demands and the capacities of link facilities is assumed. If the divisibility property does not hold, the problem will become even more difficult to solve. The algorithm combines a column generation scheme with the strong cutting plane approach. The column generation scheme is used to solve the LP relaxation of the problem efficiently. Moreover, the valid inequalities are appended to strengthen the formulation. Although more computational testing is required, the first computational results show that the proposed algorithm can solve the problem in a reasonable time and, hence, it can be used for practical purposes.

Appendix A

A.1. Proof of Theorem 1

Proof. We give a proof by showing that VPNDP has a NP-hard problem as a special case. Let us define a special case (1-VPNDP) of VPNDP by considering only one type of demand. We show the NP-hardness of 1-VPNDP by showing that the corresponding feasibility problem $\mathscr{F}$1-VPNDP is NP-complete. Notice that the NP-hardness of 1-VPNDP implies the NP-hardness of VPNDP. $\mathscr{F}$1-VPNDP can be stated as follows:

Table 9
Results for problem set 1 ($|V| = 7, |E| = 19$)

Instance	Z_{LP0}	Z_{LP}	Z_{IP}	Gap0	Gap	m	n	nCuts	tc	nit	nn	Time
1	3048.43	3303.25	3474	12.25	4.92	94	69	77	406	664	476	7.8
2	3627.27	3862.33	3975	8.75	2.83	103	78	70	340	435	294	7.68
3	4080.48	4315.72	4392	7.09	1.74	112	87	76	329	354	238	14.17
4	4302.39	4541.81	4664	7.75	2.62	121	96	88	496	2799	1532	66.35
5	4359.72	4714.83	4816	9.47	2.10	127	102	91	450	882	734	20.52
6	3393.18	3643.54	3768	9.95	3.30	210	109	75	722	314	148	9.67
7	4021.00	4224.00	4288	6.23	1.49	231	130	46	688	158	84	6.96
8	4506.32	4782.38	4946	8.89	3.31	252	151	92	1040	2752	1756	181.1
9	4762.48	4976.67	5080	6.25	2.03	273	172	77	1086	872	608	54.42
10	4828.99	4986.81	5120	5.68	2.60	287	186	83	1121	990	682	78.01

Table 10
Results for problem set 2 ($|V| = 10, |E| = 35$)

Instance	Z_{LP0}	Z_{LP}	Z_{IP}	Gap0	Gap	m	n	nCuts	tc	nit	nn	Time
1	4801.67	5349.13	5850	17.92	8.56	160	116	197	1191	2223	1164	142.18
2	5287.62	5816.12	6264	15.59	7.15	169	125	188	958	1220	814	75.14
3	5459.79	5981.04	6592	17.18	9.27	178	134	254	1270	6870	3474	582.4
4	5290.98	5979.18	6614	20.00	9.60	184	140	412	1742	9232	6874	1767.58
5	5630.77	6235.13	6948	18.96	10.26	190	146	442	1679	11,080	9882	2271.77
6	5202.09	5773.74	6214	16.28	7.08	360	176	198	2692	1466	962	299.67
7	5851.88	6355.85	6868	14.80	7.46	381	197	215	2465	2341	1156	584.59
8	6075.55	6647.61	7150	15.03	7.03	402	218	213	2334	2248	1112	574.25
9	5936.17	6699.01	7385	19.62	9.29	416	232	450	3846	22,536	15,910	14,036.08
10	6332.34	6947.42	7664	17.38	9.35	430	246	451	3521	14,814	13,538	10,941.29

Instance An undirected graph $G = (V, E)$, a set of node pairs $P = \{p_1, p_2, \ldots\}$, only one type of demand, i.e. $|T| = 1$, a set of link facility types F, a size a_t of demand type $t \in T$, a number n_p^t of demand for each $t \in T$ and $p \in P$, a capacity b_f of link facility $f \in F$, a number τ_f of ports needed for a link facility $f \in F$, a number d_v of available ports for each $v \in V$, an installation cost c_f^e for each $e \in E$ and $f \in F$, and a nonnegative integer C. All data are nonnegative integer.

Question Is there a solution of VPNDP with total installation cost at most C?

$\mathscr{F}$1-VPNDP is in NP since we can check the feasibility of any guessed solution in polynomial time. Note that, if multiple item sizes are considered, we need to solve the variable sized bin packing problem in the checking-stage, which is known to be NP-complete. Therefore, the feasibility problem of VPNDP will not be in NP if multiple item sizes are considered.

Next, we transform an arbitrary instance of the simple two-commodity integer flow problem in undirected graphs (referred to as the simple U2CIF), which is NP-complete [3], into an instance of $\mathscr{F}$1-VPNDP.

The simple U2CIF can be stated as follows:

Instance A graph $G = (V, E)$, specified vertices s_1, s_2, t_1 and t_2, and requirements $R_1, R_2 \in \mathbf{Z}_+$.

Question Are there two flow functions $f_1, f_2 : \{(u, v), (v, u) : (u, v) \in E\} \to \{0, 1\}$ such that:

1. either $f_i((u, v)) = 0$ or $f_i((v, u)) = 0$ for all $(u, v) \in E$ and $i \in \{1, 2\}$,
2. $\max\{f_1((u, v)), f_1((v, u))\} + \max\{f_2((u, v)), f_2((v, u))\} \leqslant 1$ for each $(u, v) \in E$,

3. flow f_i is conserved at v for each $v \in V - \{s_i, t_i\}$, $i \in \{1,2\}$, and
4. the net flow into t_i under flow f_i is at least R_i for $i \in \{1,2\}$?

For details, refer to Garey and Johnson [4] and Even et al. [3].

Given an arbitrary instance of the simple U2CIF, we can construct an instance of $\mathscr{F}$1-VPNDP in polynomial number of steps as follows.

Consider an arbitrary instance of the simple U2CIF given by a graph $G = (V, E)$, the four vertices s_1, t_1, s_2 and t_2, and the requirements R_1, R_2.

As shown in Fig. 5, we create a node $v(e)$ for each edge $e \in E$ and construct a graph $G' = (V', E')$, where $V' = V \bigcup \{v(e) : e \in E\}$ and $E' = \{(s_1, t_1), (s_2, t_2)\} \bigcup \{(v, v(e)) : v \in V, e \in E, e$ meets v in graph $G\}$. Then, consider the following instance of $\mathscr{F}$1-VPNDP:

$G' = (V', E')$,

$P = \{p_1, p_2\}$,
where $p_1 = (s_1, t_1)$ and $p_2 = (s_2, t_2)$,

$T = \{1\}$, $a_1 = 1$, $n_1^1 = R_1$, $n_2^1 = R_2$,

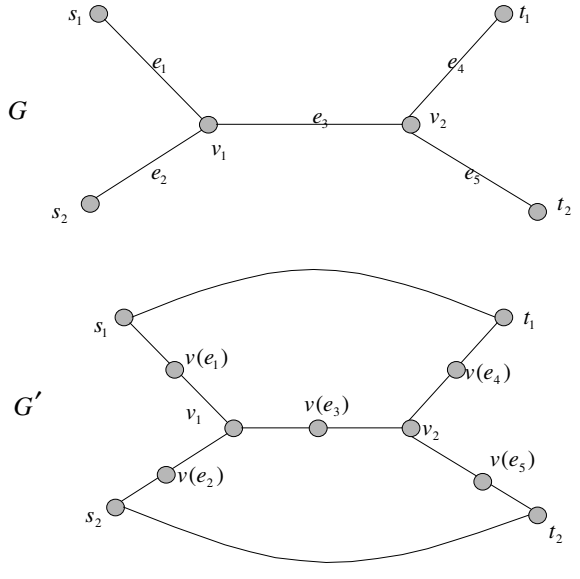

Fig. 5. Graph construction.

$F = \{1\}$, $b_1 = 1$, $\tau_1 = 1$,

$$d_v = \begin{cases} \infty & \text{if } v \in V, \\ 2 & \text{if } v \in V' - V, \end{cases}$$

$$c_1^e = \begin{cases} 0 & \text{if } e \in E' - \{(s_1, t_1), (s_2, t_2)\}, \\ 1 & \text{if } e \in \{(s_1, t_1), (s_2, t_2)\}, \end{cases}$$

$C = 0$.

In the graph G', no more than one demand can be loaded on any edge $e \in E' - \{(s_1, t_1), (s_2, t_2)\}$. Moreover, if the installation cost is 0, all demands should be loaded only on the edges in $E' - \{(s_1, t_1), (s_2, t_2)\}$. Therefore, there are two flow functions for graph G satisfying the constraints of the simple U2CIF if and only if we can create a network on graph G' carrying the demands at installation cost 0.

The theorem follows this. □

Note that we consider F and T such that $|F| = |T| = 1$ in the proof. It is easy to see that VPNDP is still NP-hard for $F = \{1,2\}$ and $T = \{1, \ldots, n\}$ such that $b_2 | b_1$ and $a_{t+1} | a_t$, $t = 1, \ldots, n-1$.

A.2. Proof of Theorem 3

Proof. Suppose there exists a vector $(\tilde{x}, \tilde{y}) \in \mathbf{P}$. Then we have $\sum_{t \in T} a_t \tilde{x}_{it}^f \leqslant b_f \tilde{y}_i^f$ for all $i = 1, \ldots, m$ and $f = 1, 2$. Since $\tilde{y}_i^f$ is a 0–1 variable, from Theorem 2 we have

$$\sum_{t \in T} \left\lfloor \frac{a_t}{a_k} \right\rfloor \tilde{x}_{it}^f \leqslant \left\lfloor \frac{b_f}{a_k} \right\rfloor \tilde{y}_i^f \quad \text{for } f = 1, 2,\ i = 1, \ldots, m \text{ and } k \in T.$$

By summing up the above inequalities, we get the following:

$$\sum_{t \in T} \left\lfloor \frac{a_t}{a_k} \right\rfloor \left(\sum_{i=1}^{m} \tilde{x}_{it}^1 + \sum_{i=1}^{m} \tilde{x}_{it}^2 \right) \leqslant \left\lfloor \frac{b_1}{a_k} \right\rfloor \sum_{i=1}^{m} \tilde{y}_i^1 + \left\lfloor \frac{b_2}{a_k} \right\rfloor \sum_{i=1}^{m} \tilde{y}_i^2 \quad \text{for } k \in T.$$

Therefore, there exists an integer vector $y \in \mathbf{D}$ such that $n_t = \sum_{i=1}^{m} \tilde{x}_{it}^1 + \sum_{i=1}^{m} \tilde{x}_{it}^2$ for $t \in T$ and $y_f = \sum_{i=1}^{m} \tilde{y}_i^f$ for $f = 1, 2$.

Conversely suppose there exists an integer vector $y \in \mathbf{D}$. Then, there exists an integer vector $(\tilde{x}, \tilde{y}) \in \mathbf{P}$ such that $n_t = \sum_{i=1}^{m} \tilde{x}_{it}^1 + \sum_{i=1}^{m} \tilde{x}_{it}^2$ for $t \in T$ and $y_f = \sum_{i=1}^{m} \tilde{y}_i^f$ for $f = 1, 2$. We can show this by generating such a vector using a modified first-fit decreasing algorithm. Let us index the bins as $1, \ldots, y_1, y_1 + 1, \ldots, y_1 + y_2$, where the size of the bin indexed by j is b_1 if $j \leqslant y_1$, otherwise, its size is b_2. Note that y_1 and y_2 are fixed here. Then, we can pack n_1 items of size a_1 assigning each of them to the lowest indexed bin in which it fits. The algorithm ends in failure if any of the items of size a_1 cannot be assigned. Otherwise, the above procedure is recursively executed for the remaining items of sizes $a_2, a_3, \ldots, a_n$.

We can show by contradiction that the algorithm always succeeds for $b_2 | b_1$ and $a_{t+1} | a_t$, $t = 1, \ldots, n-1$. Assume that the algorithm ends in failure while packing the items of size $a_{k'}$. Then, for each bin $j = 1, \ldots, y_1 + y_2$, the remaining capacity is less than $a_{k'}$ and only the items of size $a_1, \ldots, a_{k'}$ are packed. Hence, the capacity used by the items for each bin j can be expressed as follows:

$$\begin{cases} \lfloor b_1/a_{k'} \rfloor a_{k'} & \text{for } j = 1, \ldots, y_1, \\ \lfloor b_2/a_{k'} \rfloor a_{k'} & \text{for } j = y_1 + 1, \ldots, y_1 + y_2, \end{cases}$$

since $a_{k'} | \cdots | a_1$. Then, the total capacity occupied by the items is equal to

$$\left\lfloor \frac{b_1}{a_{k'}} \right\rfloor a_{k'} y_1 + \left\lfloor \frac{b_2}{a_{k'}} \right\rfloor a_{k'} y_2.$$

However, there exists an unpacked item of size $a_{k'}$. Therefore we have that

$$\sum_{t=1}^{k'} a_t n_t > \left\lfloor \frac{b_1}{a_{k'}} \right\rfloor a_{k'} y_1 + \left\lfloor \frac{b_2}{a_{k'}} \right\rfloor a_{k'} y_2.$$

However, this contradicts the assumption that $y \in \mathbf{D}$, that is,

$$\sum_{t \in T} \left\lfloor \frac{a_t}{a_k} \right\rfloor n_t \leqslant \left\lfloor \frac{b_1}{a_k} \right\rfloor y_1 + \left\lfloor \frac{b_2}{a_k} \right\rfloor y_2 \quad \forall k \in T.$$

The theorem directly follows. □

References

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance, Branch-and-price: Column generation for solving huge integer programs, Operations Research 46 (1998) 316–329.

[2] E.G. Coffman, M.R. Garey, D.S. Johnson, Bin packing with divisible item sizes, Journal of Complexity 3 (1987) 406–428.

[3] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM Journal on Computing 5 (1976) 691–703.

[4] M.R. Garey, D.S. Johnson, Computers and Intractability: A guide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.

[5] ITU-T recommendation E.735, Framework for traffic control and dimensioning in B-ISDN, 1997.

[6] J. Kang, S. Park, Algorithms for the variable sized bin packing problem, European Journal of Operational Research 147 (2003) 365–372.

[7] M. Laguna, F. Glover, A tabu search approach, Management Science 39 (1993) 492–500.

[8] T.L. Magnanti, P. Mirchandani, R. Vachani, Modeling and solving the two-facility capacitated network loading problem, Operations Research 43 (1995) 142–157.

[9] O. Marcotte, The cutting stock problem and integer rounding, Mathematical Programming 33 (1985) 82–92.

[10] G.L. Nemhauser, S. Park, A polyhedral approach to edge coloring, Operations Research Letters 10 (1991) 315–322.

[11] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, 1988.

[12] K. Park, S. Kang, S. Park, An integer programming approach to the bandwidth packing problem, Management Science 42 (1996) 1277–1291.

[13] B.H. Ryu, H. Ohsaki, M. Murata, H. Miyahara, Design algorithm for virtual path based ATM networks, IEICE Transactions on Communications E79-B (1996) 97–107.

[14] T. Wu, N. Yoshikai, ATM Transport and Network Integrity, Academic Press, 1997.

[15] J.Y. Yen, Finding the K shortest loopless paths in a network, Management Science 17 (1971) 712–716.